T.C. İSTANBUL KÜLTÜR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SANAL HEYKELTIRAŞLIK SİSTEMLERİ İÇİN YENİ DIŞ ÇEPER VE DALLANMA YAKLAŞIMLARININ GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

EMRE GÖCEN

Anabilim Dalı: Bilgisayar Mühendisliği

Programı: Bilgisayar Mühendisliği

Tez Danışmanı: Yard. Doç. Dr. Kemal YÜKSEK

ÖNSÖZ

Tez çalışmam süresince sonsuz sabır gösterip hiçbir zaman benden yardımlarını esirgemeyen tez danışmanım Sayın Yard. Doç Dr. Kemal YÜKSEK'e, uzun bir sürece yayılan tez çalışmam boyunca benden desteğini esirgemeyen aileme, sonsuz teşekkürü bir borç bilirim.

Emre GÖCEN

Haziran 2011

İÇİNDEKİLER

KISALTMALAR

CSG	: Constructive Solid Geometry (Yapısal Katı Geometri)
NC	: Numerical Control (Sayısal Kontrol)
CNC	: Computer Numerical Control (Bilgisayar Sayımlı Kontrol)
VOXEL	: Volumetric Pixel
DEXEL	: Depth Element

TABLO LİSTESİ

ŞEKİL LİSTESİ

Üniversitesi : **İstanbul Kültür Üniversitesi**

Enstitüsü : **Fen Bilimleri**

Anabilim Dalı : **Bilgisayar Mühendisliği**

Programı : **Bilgisayar Mühendisliği**

Tez Danışmanı : **Yard. Doç. Dr. Kemal YÜKSEK**

Tez Türü ve Tarihi : **Yüksek Lisans – Haziran 2011**

ÖZET

SANAL HEYKELTIRAŞLIK SİSTEMLERİ İÇİN YENİ DIŞ ÇEPER VE DALLANMA YAKLAŞIMLARININ GELİŞTİRİLMESİ

Emre GÖCEN

Bu çalışmada, sanal heykeltıraşlık sistemlerine yeni alternatif bir çözüm önerilmiş ve gerçeklenmiştir. "Sanal düzlem" tabanlı bu yaklaşım ile sistemin önemli iki unsuru olan dallanma işlemi ile objelerin işlenmesi safhaları başarılı bir şekilde tamamlanmıştır. Oluşturulan yapı dış çeperlerin sağlıklı ve hızlı oluşturulması ile de sistem entegre edilmiştir. Temel geometrik özelliklerin uygun bilgisayar veri yapıları ile desteklenmesi sistemin gerçeklenmesinde önemli rol oynamıştır. Katı Model ve işleyicinin şeklinden bağımsız esnek yapısı ile sistem, hızlı ve az saklama alan ihtiyacı gerektiren yapısı ile mevcut dexel ve voxel çözümlerine üstünlük sağlamıştır. Çalışmanın, bilinen örnek problemleri aşma yeteneği ve performans değerleri incelenmiş ve mevcut yaklaşımlar ile kıyaslanmıştır. Sistemin gerçeklenmesinde, .Net platformu üzerinde C# programlama dili ve OpenGL grafik kütüphaneleri kullanılmıştır.

Anahtar Kelimeler : **Katı Hacim Modelleme, Sanal Heykeltıraşlık, Yeniden İnşa, Dexel, Voxel, Örgüleme**

Bilim Dalı Sayısal Kodu : **215.03.00**

University : **İstanbul Kültür University**

Institute : **Institute of Science**

Science Programme : **Computer Engineering**

Programme : **Computer Engineering**

Supervisor : **Asst. Prof. Dr. Kemal YÜKSEK**

Degree Awarded and Date : **MS – June 2011**

SUMMARY

DEVELOPING A NEW CONTOURING AND BRANCHING APPROACH IN COMPUTER BASED VIRTUAL SCULPTING

Emre GOCEN

In this study, A new alternative solution for the virtual sculpting systems has been proposed and developed. Branching and processing solid objects which are two important phases of the system have been successfully completed with this "virtual plane" based system. Produced platform has been integrated with the rapid and rebost construction of the contours. Supporting the basic geometric properties with the suitable computer data structures has played an important role for the implementation of the system. The system has lots of advantage against voxel and dexel volume modeling cause of its solid model and virtual tool independent structure. The system is faster, needs less drive capacity and resume less memory with that structure. Ability of solving known sample problems and advantages in comparison with actual approaches with regard to performance values of study are shown. In the implementation of the system, .Net C# programming language and OpenGL graphics libraries are used.

Keywords : **Solid Volume Modeling, Virtual Sculpting, Reconstruction, Dexel, Voxel, Branching**

Science Code : **215.03.0**

1. GİRİŞ

Sanal heykeltıraşlık, kullanıcıların sanal bir çalışma ortamında 3 boyutlu geometrik modelleri, bilgisayar ekranından görüp, gerçek hayatta bir heykeltıraşın bir nesneyi işlemesine benzer şekilde etkileşimli ve gerçek zamanlı olarak işleyebilmesidir [1,2]. Böyle bir sistem, kullanıcılara standart geometrik şekillerden bağımsız serbest formda modelleri işleyebilme imkânı ile sınırsız bir tasarım gücü sağlar.

Sanal heykeltıraşlık sistemleri, temelde üç aşama üzerine kuruludur:

1. Dış cephe hatlarının belirlenmesi (Contouring),
2. Belirlenen dış cephe hatlarının işleyici aletle etkileşime girmesi (Sculpting),
3. Modelin, işleyici aleti ile etkileşiminden sonra yeniden inşası (Reconstruction).

Teorik olarak iki tür sanal heykeltıraşlık sistemi bulunmaktadır. İlk sistem, yüzeyi temel almaktadır ve değiştirebilen geometrik nesne modellemesinden türetilmiştir. Bu sistemlerde modelleme, yüzey üzerindeki eğriler veya tepe noktaları esas alınarak yapılabilmektedir.

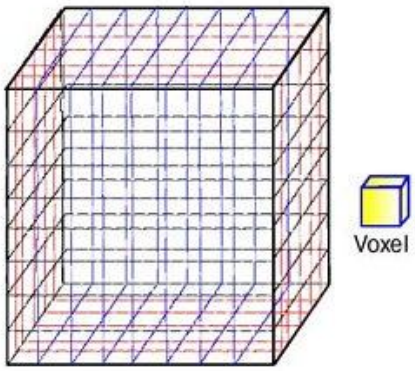
Diğer sanal heykeltıraşlık sistemindeki modelleme ise, sayısal kontrollü simülasyon (NC Simulation) işlemleri ile benzerlik gösterir ve nesnenin, sanal bir alet ile etkileşimi sonrasında kendisinden parçaların kopması esasına dayalı olan hacimsel modellemedir.

Hacimsel modellemede, gerçek anlamda cismin iç ve dış geometrisinin tanımı yapılmış olur. Yüzey modelleme yöntemlerinin zayıf kaldığı birçok nokta bu yöntemde giderilmiştir. Katı modellemenin esas özelliği görüntünün ötesinde cismin iç ve dış geometrisinin bilgi kütüğü şeklinde bilgisayara geçmiş olmasıdır. Modelin hacim, moment, ağırlık gibi fiziksel özellikleri hakkında bilgi edinilebilir ve kesitler alınarak cismin iç geometrik formu incelenebilir. Cisimlerin yüzeylerindeki renkler, geçirgenlik, ışık yoğunluğu ve gölgeleme yapılabilir.

Hacim şekillendirilmesi, kullanıcıların kontrolündeki kesici aletin modelle etkileşime girerek, nesneden parçaları koparmasıyla mümkündür. Bu süreç, kullanıcın istediği şekli elde etmesine kadar devam eder. Üç boyutlu verilerin gerçek zamanlı işlenmesinde ve CNC (Computer Numerical Control) işlem simülasyonlarındaki avantajları nedeni ile hacim modellemesi, sanal heykeltıraşlık sistemlerinde büyük rağbet görmektedir. Sanal heykeltıraşlığın bu türünde en yaygın kullanılan hacim modellemeleri, kolay anlaşılabilir ve uygulanabilirlikleri nedeniyle, voxel ve dexeldir. Hacim modellemeleri bu iki yapı üzerine kuruludur [3].

Voxel (Volumetric ve Pixel kelimelerinin kısaltması olarak geçmektedir), hacimsel modellemenin bir çeşididir. Bu modelleme adını, nesnelerin modellendiği en küçük birim olduğu kabul edilen ve voxel adı verilen küçük küplerden almaktadır [4].

Şekil 1.1'de görüldüğü üzere bu küpler üç boyutlu uzayda, sanal bir ızgara üzerindeki birim elemanı temsil etmektedir [5]. Bu durum iki boyutlu resim dosyaları için birim elemanı simgeleyen pixel ile benzeşmektedir. Bir resim dosyası için pixel ne ise, üç boyutlu bir nesne modellemesi için de voxel onu simgelemektedir.



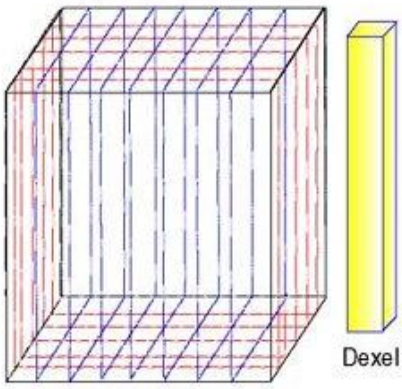
Şekil 1.1: Voxel Yapısının Temsili Gösterimi

Bu modellemede nesneler küplerle (voxel) oluşturulur ve en küçük düzenlenebilir parçalardır. Bu yapı, tıpkı legolardaki yapıya benzetilebilir. Küçük parçalar birbirine eklenerek daha büyük objeler oluşturulmaktadır.

Voxel modellemesinde nesneler parçalanamaz bir birim olan küplerden meydana geldiğinden dolayı, işleyici alet ile nesnenin herhangi bir etkileşimi nesneden küpleri koparacaktır. Dolayısıyla bu durum işleyici aletin, küplere en ufak teması sonrası dokunulan küplerin kopmasına neden olacaktır.

İşleyici aletin, küpün sadece bir köşesine temas etmiş olsa dahi, küpün tamamını koparacak olması, nesnenin sanal heykeltıraşlık işlemlerine maruz kaldıktan sonra daha az keskin ve gerçek hayattaki görünümün aksine pürüzlü bir şekilde görünmesine sebep olacaktır.

Bu sorunu ortadan kaldırabilmek amacı ile, küplerin oluşturduğu voxel birim elemanları yerine doğru parçalarından oluşan dexellerin kullanıldığı dexel hacim modeli ortaya atılmıştır. Bu hacim modellemesinde, işleyici aletin temas ettiği yerlerin sütundan kopması söz konusudur.



Şekil 1.2: Dexel Yapısının Temsili Gösterimi

Dexel yaklaşımı voxele oranla daha keskin bir nesne işleme imkânı verebilmektedir. Buna karşın dexel olarak isimlendirilen doğru parçalarının sayısı işleme kalitesi ile doğru orantı gösterirken aynı orantı ve oluşan işlem yüküne de yansır. Ayrıca bazı özel geometriler yeniden inşa esansında nesne anomalilerine de sebep olabilmektedir. Her iki yaklaşımın dezavantajlarını ortadan kaldıran bir yaklaşım tez çalışmasının ana konusu olarak ortaya çıkmıştır.

Bu çalışmada voxel ve dexel hacim modelleri yerine "sanal düzlem"lerle belirlenmiş dış çeperler üzerinde yeniden inşa algoritmasının geliştirilmesi üzerinde durulacaktır. Asıl hedef ise sanal heykeltıraşlık sistemi için daha keskin ve doğru sonuçlar verebilen bir algoritma oluşturup, bu sistemi olabildiğince hızlandırmaktır.

Sanal düzlemler kullanılarak, öncelikle dexel modeli ile hedeflenen işlenecek birim eleman sayısını düşürme işlemi daha başarılı bir şekilde gerçeklenmiştir. İşlenecek daha az sayıdaki birim eleman, algoritmanın daha az işlem yapması ve sistem kaynaklarının o denli daha düşük kullanılması anlamına geldiğinden "sanal düzlem" kavramı sistem için çok önemli bir yer tutmaktadır.

İşlenecek birim eleman sayısının performansa etkisi bu tez çalışmasının ileriki bölümlerinde yapılacak performans testleri belirtilecektir.

Sanal heykeltıraşlık sistemi ilk olarak Galyean tarafından ortaya atılmıştır. Doksanlı yılların başlarında ortaya atılan bu fikir, uygulama aşamasında tatmin edici sonuçlar verememiş olsa bile, bu düşüncenin pratiğe dökülmesinin imkânsız olmadığını gösterecek kadar başarılıdır [6].

Voxel hacim modelleri ile oluşturulmuş nesneleri üç boyutlu sanal ortamda işleyici bir aletle istediği şekle getirmeyi amaçladığı bu sistem voxel yapısı nedeniyle istenildiği kadar keskin ve doğru modelleme yapılabilme imkânı sağlayamamaktaydı. Sistem kaynaklarını da oldukça fazla tüketen bu yapı her şeye rağmen bu fikrin gerçekleştirilebilir olacağını düşündürecek kadar başarılıdır.

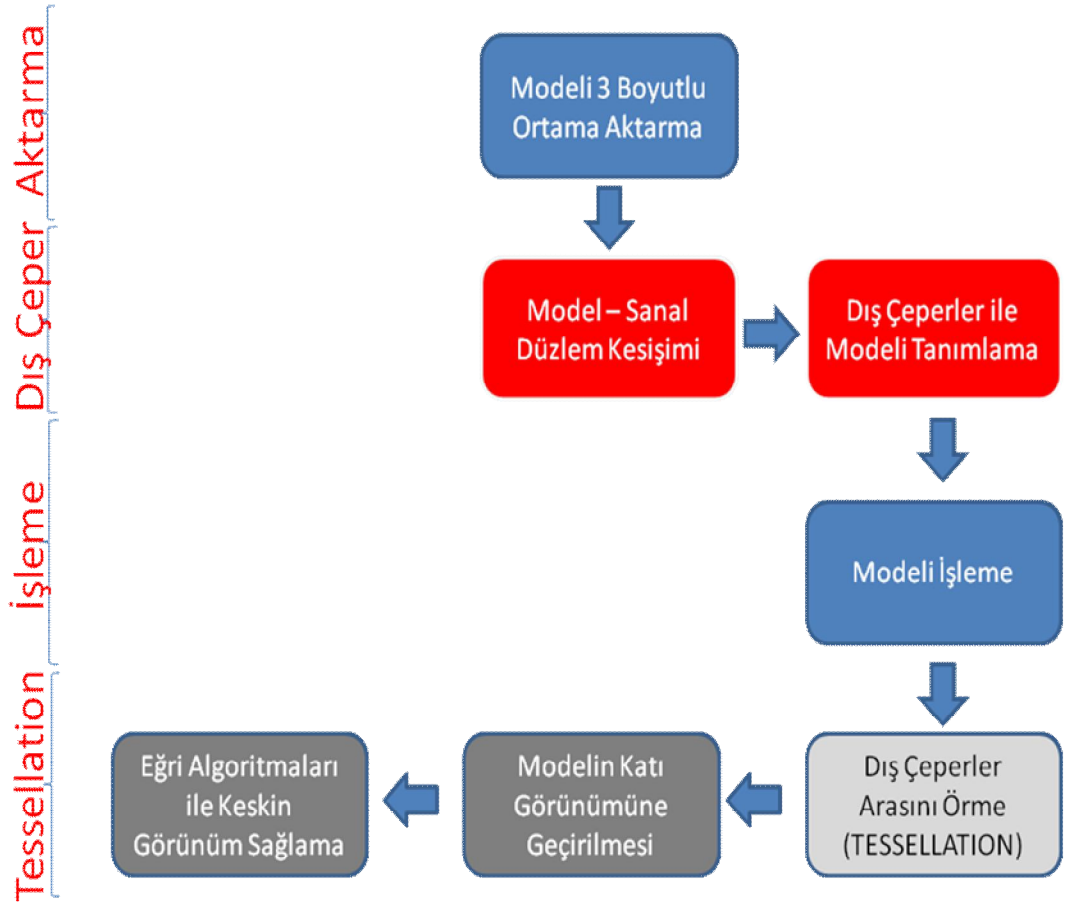
İlerleyen dönemlerde yüzeysel ve hacimsel metodlar kullanılmaya başlanmıştır. Yüzeysel yapı ile sistemi kurmaya çalışan McDonnell [7]. tri-parametrik BSpline yüzeyleri kullanmıştır Ancak sistem, kontrol noktalarına bağımlılığı nedeniyle, son kullanıcılara gerektiği gibi sezgisel modelleme yapma imkânı vermemiştir.

Hacimsel modelleme bazında ise Mizuno [8], CSG(Constructive Solid Geometry) hacim gösterimini esas alarak birtakım çalışmalar yapmıştır. Ancak algoritmanın heykeltıraşlık işlemleri esnasında ihtiyaç duyduğu aşırı hesaplamalar, sistemin performansını çok fazla düşürmüştür.

Galyean ve Hughes'un [9]. üzerinde çalıştığı bir başka hacim modellemesini temel alan sanal heykeltıraşlık sistemi ise bir türlü istenen keskinlikte modellemeler yapılmasına olanak tanımamıştır Ancak Galyean ve Hughes diğer sistemlerin aksine, kendi sistemlerinde model üzerine yapılan ekleme, çıkarma işlemlerinin yanında renklendirme, zımpara kağıdı gibi araçlar da eklemişlerdir. Wang ve Kaufman sistemlerine daha karmaşık ve gelişmiş araçlar eklemişlerdir [10].

Ferley, sanal heykeltıraşlık sisteminde voxel modeli üzerinde yoğunlaşmış ve bir voxel küpünü 27 parçaya bölerek hiyerarşik yapı kurarak daha keskin modelleme yapmayı amaçlamıştır [11]. Çünkü voxel parçacıkları küçüldükçe kopacak parçalarda o derece küçüleceğinden daha keskin görüntüye ulaşılacaktı. Fakat sistemin aşırı bellek tüketimi ve işleyici aletin sadece elips şeklinde olması sitemin en büyük dezavantajı olarak göze çarpmaktadır.

Modelleme seçimi dışında, işleyici alet ile işleme giren nesnenin bu işlemlerden sonra yeniden inşası da bir diğer önemli araştırma konusunu teşkil etmektedir. Bu çalışmada dallanma(branching) ve örgüleme(tessellation) alt aşamaları ile belirlenen bu süreçte; birbirine paralel olarak sıralanmış sanal düzlemlerin doğru şekilde birleştirilip modellenmesi gerçeklenmektedir (Şekil 1.3).



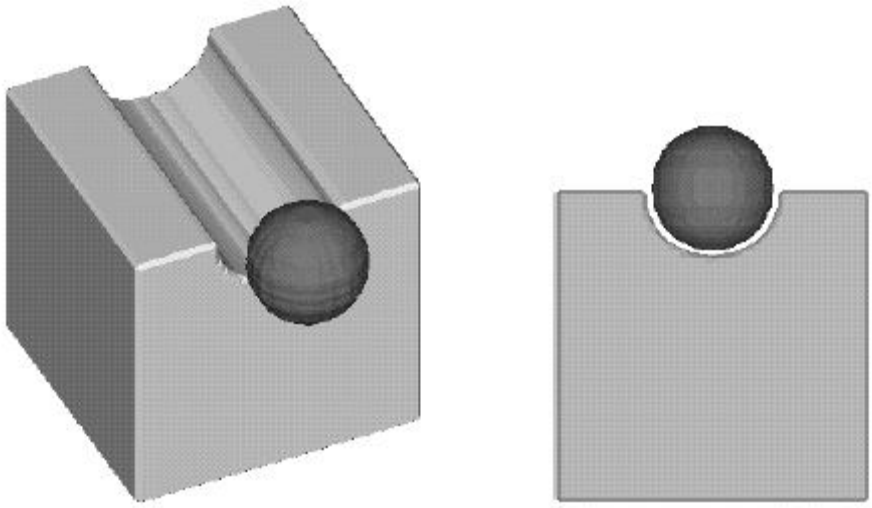
Şekil 1.3: Sistemin Çalışma Süreci

Şekil 1.3 ile görüldüğü üzere sanal heykeltıraşlık sisteminde esas olan unsurlar işlenecek nesne modeli ile işleyici alettir. Sistemde yapılan ilk işlem, modelin koordinatları kullanılarak işlenecek nesnenin üç boyutlu sanal ortama aktarılmasıdır. Bu süreçte model, her üç parametre xyz uzayında bir noktayı belirtmek koşulu ile bir dizi üçgene ait köşe koordinatları ile tanımlanır.

Sanal ortama aktarılan model işleyici alet ile etkileşime girecektir. Bu etkileşim için öncelikle işleyici alet ve işlenecek nesne modelinin dış çeperler ile tanımlanması gerekmektedir [12]. Böylece aynı tanımlamaya sahip iki nesnenin etkileşimi daha rahat bir şekilde hesaplanacaktır. Bu kolaylığı sağlayabilmek için nesneler üzerine bir dizi sanal düzlem yollanır. Sanal düzlemler ile nesnelerin kesişmesinden nesnelere ait dış çeperler elde edilir [13].

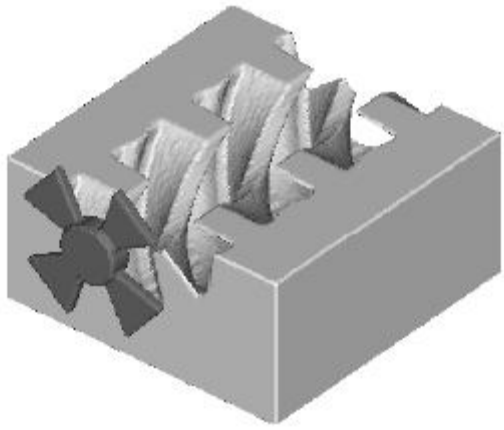
Kesişim algoritması ile dış çeperlerin elde edilmesi aşamasında dikkat edilmesi gereken en önemli unsurlardan bir tanesi, nesne üzerine gönderilecek sanal düzlemlerin sayısı ve sıklıklarıdır. Çok fazla sanal düzlem göndermek bir yandan nesneyi daha detaylı tanımlarken, bir yandan da daha fazla işlem yapılmasına neden olacaktır. Benzer şekilde çok az sanal düzlem göndermek çeperler arasındaki mesafeyi arttıracağından modelin yeniden inşası sırasında büyük problemler çıkarabilirken algoritmanın çok daha az hesaplama yapmasını sağlayacaktır. Dolayısıyla bu iki parametreyi dikkatli bir şekilde belirlemek gerekmektedir.

Dış çeperler ile tanımlanan işleyici alet ve model için bir sonraki aşama işleme aşamasıdır. Burada işleyici alet model üzerinden parça koparabilir veya parça ekleyebilir [8] [14]. En önemli ayrıntı, işleyici aletin şekline göre model üzerinde değişikliklerin meydana gelmesidir [15]. Örneğin alet oval bir şekle sahipse model üzerine herhangi bir etkide bulunduğunda oval darbeler vermesi gerekmektedir (Şekil 1.4).



Şekil 1.4: İşleyici Aletin Model Üzerine Etkisi

Benzer şekilde, işleyici alet köşeli bir yapıya sahipse, model üzerinde de köşeli darbeler vermelidir (Şekil 1.5). İşleme süreci sanal heykeltıraşlığın doğası gereği son kullanıcının sezgisel yeteneklerini birebir yansıtmasına olanak sağlayacak şekilde yapılanmalıdır. Kullanıcılar modele uyguladıkları işlemleri anlık ve doğru bir şekilde görmelidirler.

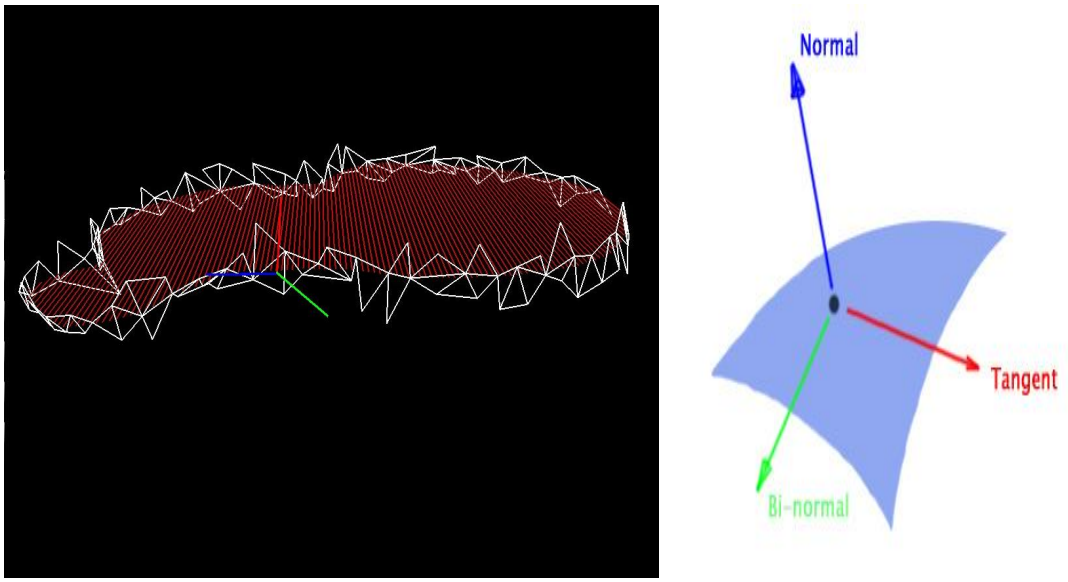


Şekil 1.5: İşleyici Aletin-Model Etkileşimi

Modelin istenen şekle getirildikten sonra dış çeperlerin örülmesi ve yeniden kapalı hacim modeline dönüştürülmesi gerekmektedir [16]. Dış çeperler arasındaki boşlukların kapatılıp modeli yeniden hacimlendirme işlemine örgüleme denmektedir. Bu tez çalışması, sanal heykeltıraşlık sistemlerinin bu aşamasını da konu almaktadır.

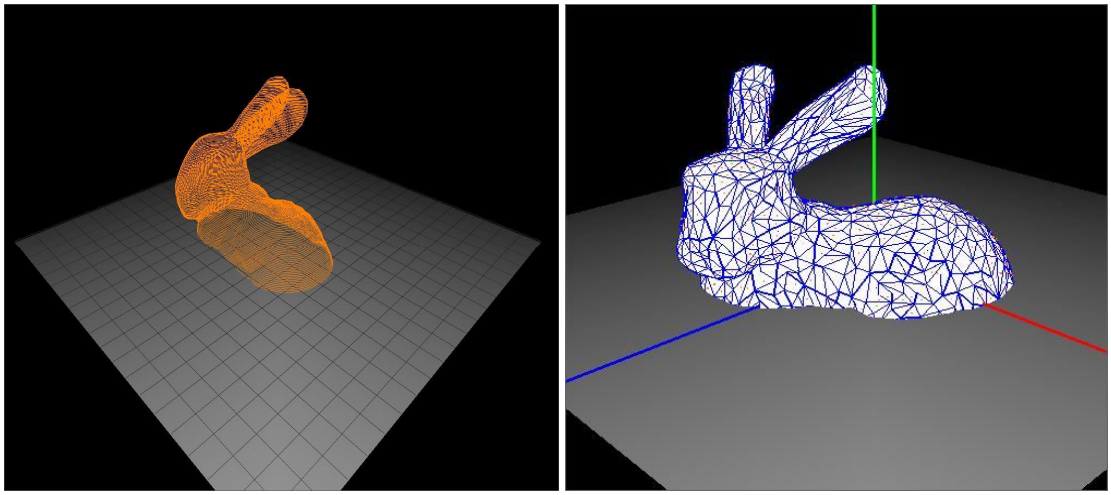
Örgüleme işleminde en önemli zorluk dış çeperler arasındaki boşlukların doğru bir şekilde doldurulmasıdır [17]. Çünkü işlenen modelin yapısı ilk başta tanımlanan yapısına göre büyük farklılıklar gösterecektir. Dolayısıyla yeni yapının dış çeperlerinin doğru eşleşme ile üçgenlerle örtülmesi bu çalışmanın aşmayı hedeflediği sorun olarak göze çarpmaktadır [18].

Bu sorunu çözebilmek için dış çeper olarak adlandırdığımız poligonları oluşturan doğru parçalarının her birine ait normal ve normale dik vektörleri kullanılmaktadır (Şekil 1.6). Bu vektörlerin yönleri, hangi iki çepere ait hangi noktalar arasında üçgenleştirme işlemi yapılacağının tespit edilmesinde büyük rol oynarlar. Kesişim algoritmasında olduğu gibi üçgenleştirme algoritmasında birtakım kıstaslar belirlenir ve bu kıstaslara göre doğru nokta eşleştirilmeleri yapılarak üçgenlerle dış çeperlerin aralarında örme işlemi yapılır.



Şekil 1.6: Normal ve Normale Dik Vektörler

Dış hatlar arası örme işlemi sonrası nesne katı model görünümüne dönüştürülür (Şekil 1.7). Böylece model işleyici alet ile işlenip tekrar ilk baştaki katı model görünümüne sahip bir şekilde ekrana yansıtılacaktır. Ancak buradaki en büyük fark, ilk başta ekrana yansıtılan modelin bu sefere işlenmiş haliyle ekrana yansıtılıyor olmasıdır.



Şekil 1.7: Dış Çeperler ve Katı(Solid) Olarak Temsil Edilen Tavşan(Bunny) Modeli

İşlenmiş nesne modeline yapılabilecek son işlem görünümü iyileştirme ve keskinleştirme üzerinedir [19]. Birtakım eğri algoritmaları kullanılarak işlenmiş haldeki modelin daha keskin görüntüye kavuşması sağlanabilmektedir. Bezier eğrileri bu işlem için yaygın olarak kullanılmaktadır [20].

"Birbirine paralel olmayan iki düzlem, bir doğru boyunca kesişirler" ilkesinden hareketle "sanal düzlemler" kullanılarak elde edilen dış çeperleri, sanal heykeltıraşlık işlemleri esnasında etkili ve doğru bir şekilde yeniden inşa edebilmek amacı ile hazırlanmış bu tez çalışması şu şekilde düzenlenmiştir:

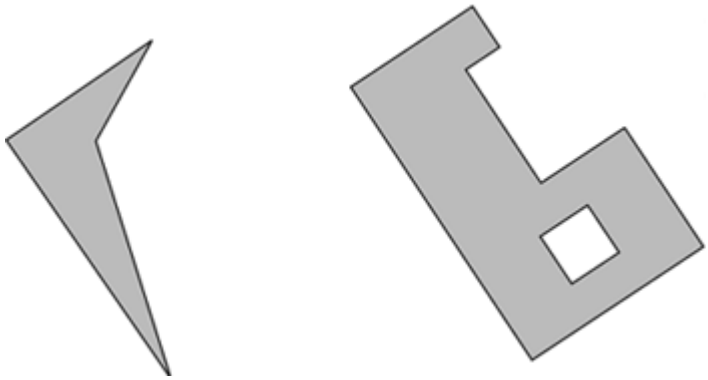
İzleyen bölümde, sanal heykeltıraşlık sistemlerinde kullanılan yeniden inşa yöntemleri irdelenip eksiklikleri ortaya konulmuş ve genel olarak dallanma ve örgüleme yapısının sanal heykeltıraşlık sistemlerindeki yeri belirtilmiştir.

3. Bölümde ise, önerdiğimiz metoda gelene kadar, sistemde gerçekleşen olaylar sıralanmış ve sistemin işleme süreci öncesindeki aşamaları sıralanmıştır. 4. Bölümde, geliştirdiğimiz metodun çalışması aşama aşama anlatılarak, yazılımın çalışma sürecinde ne gibi işlemler yaptığı, hem görsel hem de yazılı olarak detaylandırılmıştır. Bu bölümde uygulama ile ilgili birçok konu, resimli olarak ifade edilmiştir. 5. Bölümde, diğer metodlarla kıyaslamalar yapılıp, geliştirilen algoritmanın etkinliği irdelenmiştir. 6. Bölümde ise, sonuçlar ve algoritma ile ilgili gelecekte yapılması muhtemel çalışmalara yer verilmiştir.

2. DALLANMA(BRANCHING)

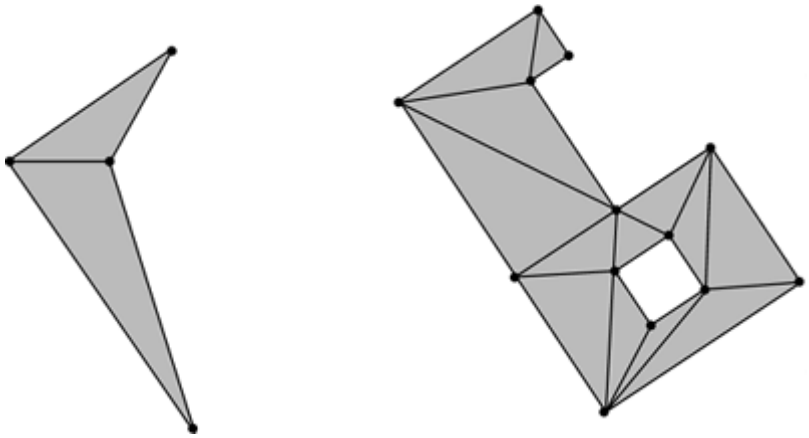
Dış hatların yeniden inşası, sanal heykeltıraşlık sistemleri için önemli bir çalışma alanını temsil etmektedir. Dallanma ve örgüleme başlıkları altında genelleyebileceğimiz bu yapı, bu tez çalışmasında dış hat çeperler arasındaki boşluğu bir örgü şeklinde kapatarak nesne modellemesini, sanal heykeltıraşlık işlemleri sonrası yeniden oluşturmak şeklinde tanımlanmaktadır.

OpenGL, DirectX gibi kütüphaneler olabildiğince hızlı kullanabilmek için geometrik ilkellerin dışbükey olması gerekmektedir. Çünkü bu tarz kütüphaneler genelde dışbükeyler için optimize edilmişlerdir. Ancak üç boyutlu ortama aktarılacak her modelin dışbükey yapıda olması beklenemez. İçbükey ve karmaşık modeller üzerinde çalışmalar yapılabilir (Şekil 2.1). Bu tarz sorunların önüne geçebilmek için karmaşık geometriye sahip şekilleri ya da bir içbükey şekli daha küçük dışbükey geometrik şekillere bölebiliriz. Üçgenler bu işlem için en yaygın olarak kullanılan ilkeldir.



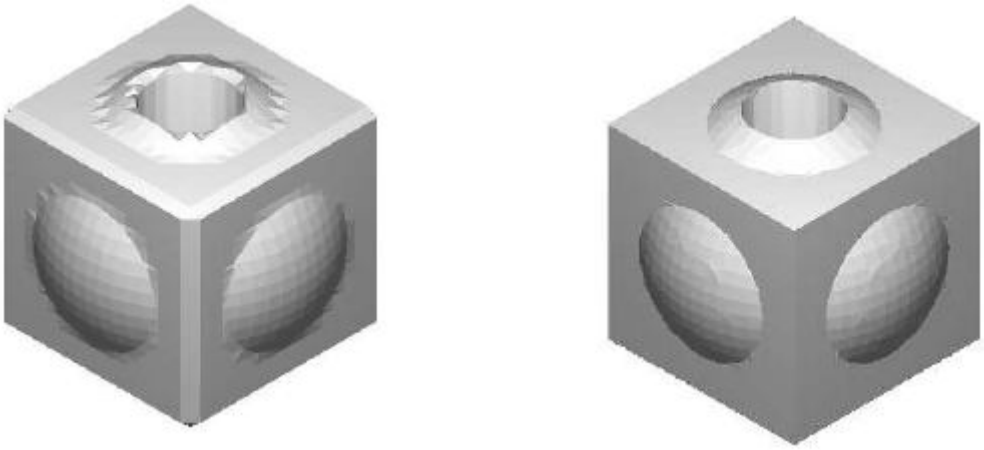
Şekil 2.1: İçbükey ve Karmaşık Yapıda İki Örnek Şekil

Sanal heykeltıraşlık sistemlerinde en yaygın olarak kullanılan primitif şekli, üçgendir. Üçgenler, karmaşık veya içbükey yapıdaki şekli, birleşerek yeniden oluştururlar (Şekil 2.2).



Şekil 2.2: Üçgenlerle Yeniden Oluşturulan Şekiller

Örgüleme işlemi esnasında üçgenlerin yerleşimi, modellenen objenin görünümü açısından oldukça önemlidir. Algoritmada yapılacak bir hata modelin dış yüzeyinin çıkıntılı olarak ve kötü bir şekilde görülmesine neden olacaktır (Şekil 2.3). Üçgenler arası boşluk bırakılmaması ve bir sıra boyunca aynı üçgen örnekleri ile örgü işleminin yapılması bu aşamanın esasları arasında yer almaktadır.



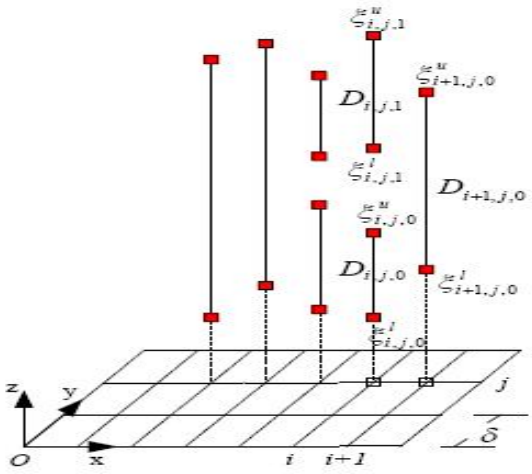
Şekil 2.3: Örgüleme Yapısının Model Görünümüne Etkisi

Ren, Zhu ve Lee, sorunsuz bir örgüleme için nesne modellemesini Tri-Dexel yapısına entegre ederek bu sorunu çözmeye çalışmışlardır [21]. Su geçirmez (water-tight) polihedral yüzey olarak tanımladıkları yapıya en uygun örgüleme işlemini yapabilmek için modeli önce Tri-Dexel moduna daha sonra da Tri-Dexel modundan polihedral yüzeye geçirmektedirler.

Çalışmalarının başlıca amaçları:

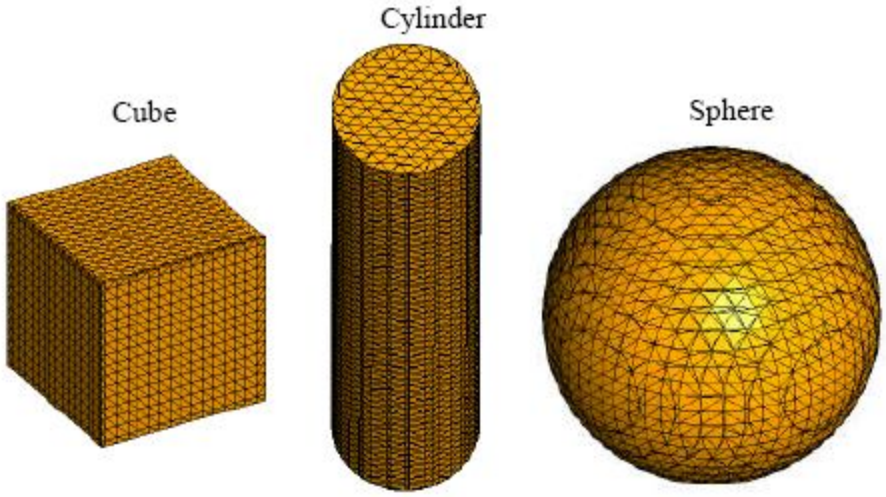
- Modelin, konum verilerini korumak,
- Kenarlar ve köşeler gibi hassas geometrik yapıları yeniden ve daha keskin olarak oluşturmak,
- Sanal heykeltıraşlık işlemleri sonucu elde edilen yeni ve işlenmiş modelin su-geçirmez bir yapıda yeniden örülmesini sağlamak,
- Ekleme ve çıkarma işlemlerini daha hızlı yapabilmek için Boolean işlemlerini kolaylaştırmak,

Tüm bunları gerçekleyebilmek amacıyla voxel nesne modellemesine alternatif olarak ortaya atılmış dexel modellemesine de yeni bir yaklaşım getirerek tek yöndeki dexelleri hem X hem de Y koordinatları üzerinden oluşturarak dexeller üzerinde daha fazla kontrole sahip olmuşlardır (Şekil 2.4).



Şekil 2.4: X ve Y Koordinatlarında Tanımlı Dexel Yapısı

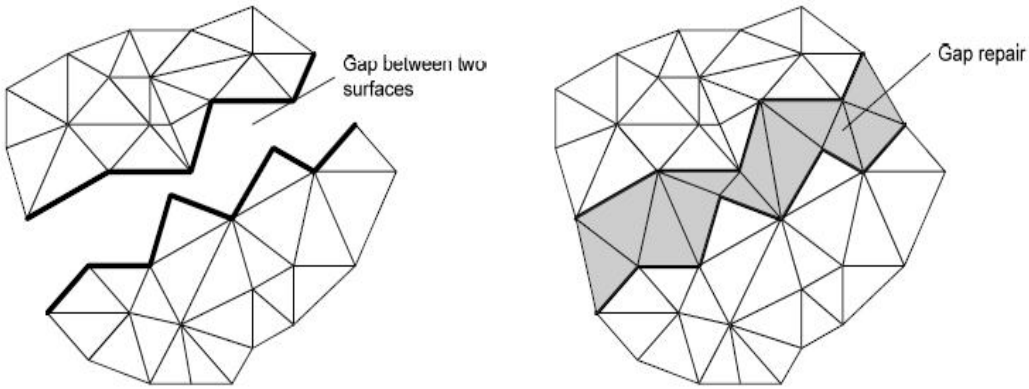
Tri-Dexel yapısının bir diğer özelliği örgüleme işlemleri için sorun oluşturan içbükey şekiller için de etkili sonuçlar verebilmesidir. Tri-Dexel yapısı üzerinde üçgen primitifleri kullanılmaktadır (Şekil 2.5). Üçgenlerle tanımlanan üç boyutlu nesne modelleri daha sonra polihedral yüzeylere dönüştürülürler.



Şekil 2.5: Üçgen Primitifleri ile Tanımlı Nesne Modelleri

Tüm artılarına rağmen tri-dexel modeli yapısı gereği, işlemler esnasında fazla birim eleman kullanmaktadır. Ayrıca üçgenler arası boşluklar ve tekrar eden örnekler(sample) görünümün bazı karmaşık modeller için yetersiz seviyede keskinlikte olmasına neden olmaktadır.

Örgüleme üzerine yapılan araştırmalarda çözülmesi üzerine en çok çaba sarfedilen problem yüzeyler arasındaki boşluğu doğru şekilde doldurabilmektir (Recover Gap Holes) [15]. Bu problem ilk olarak 90lı yılların başında Berequet tarafından tespit edilmiştir. Problemi çözmek için NURBS yüzeyleri kullanılmıştır. 2003 yılında ise Yau, STL modelini kullanarak çok daha etkili bir çözüm üretmiştir [22]. Genel olarak bakıldığında geçmiş boşluk doldurma (gap hole) çalışmalarında kullanılan algoritmalar boşlukları otomatik olarak tamir etmektedir. Ancak algoritmalar sorunu çözmeden önce, boşlukları tespit etmek için bir başka algoritma daha çalıştırmaktadırlar. Böylece sistem üzerine biriken hesaplama yükü daha da artmaktadır (Şekil 2.6).



Şekil 2.6: Boşluk Doldurma İşlemi

Genel olarak bakıldığında örgüleme işlemi, işlenen modelin tekrar inşası için önemli bir yer teşkil ettiği söylenebilir. Dolayısıyla bu aşama da kullanılacak algoritmanın, sisteme fazladan hesaplama yükü vermemesi göz önünde bulundurulmalıdır.

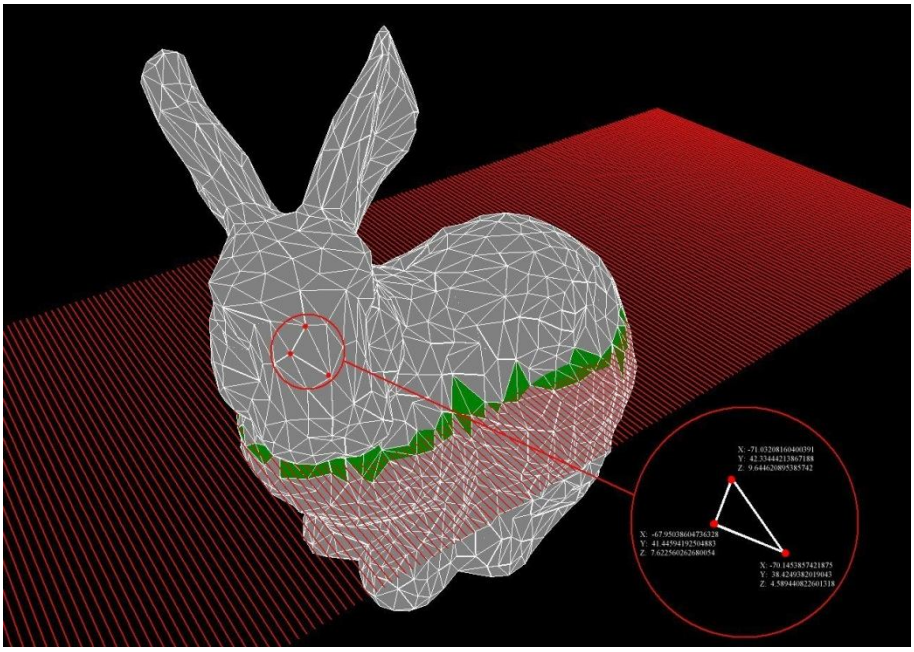
3. ÖN BİLGİLER

Bu bölümde işleyici alet ile şekillendirilen üç boyutlu modelin yeniden inşa kısmına gelmeden önceki süreci anlatılacaktır. Üçgenler ile objenin modellemesinden başlayan bu süreç, sanal düzlemlerle modelin kesişmesi sonucu elde edilen dış çeperlerle nesnenin tanımlanmasına kadar detaylandırılacaktır.

Bu bölümden sonra, tezin asıl konusu olan sanal heykeltıraşlık işlemleri sonrası modelin yeniden inşası konusu ele alınacaktır.

3.1 Üçgenlerden Model Oluşturma

Uygulama ile dış hatları belirlenecek nesne, çalışma alanına, bir metin dosyasından okunan üçgen koordinatları aracılığı ile aktarılmaktadır (Şekil 3.1.1). Tüm bu üçgen koordinatları, OpenGL içindeki özel bir yapı olan görüntü listesi(display list) ile ekran kartı üzerinde saklanıp işlemlerin ekran kartına da paylaştırılması sonucu nesnenin hızlı bir şekilde çizdirilmesi sağlanır.



Şekil 3.1.1: Modeli Oluşturan Üçgenlerin Koordinat Yapıları

Görüntü listesi, içerisinde saklı tuttuğu bir grup OpenGL komutunun, sadece kendisinin çağırılmasıyla tekrar tekrar kullanılabilmesini sağlar. En büyük etkisini performans üzerinde gösterir.

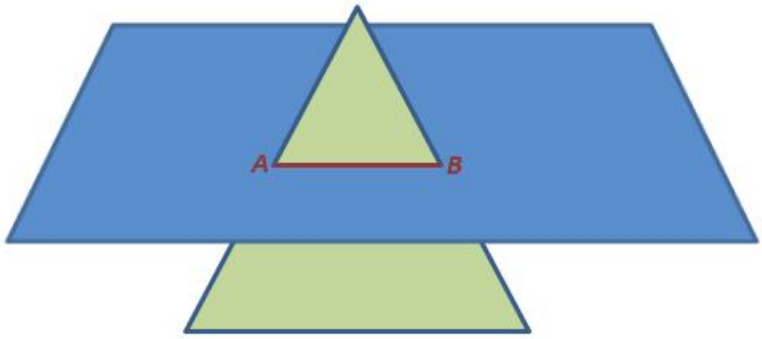
Her bir üçgen koordinatı okunarak oluşturulan model, sanal heykeltıraşlık işlemlerine tabi tutulmak üzere sanal ortama aktarılır. Bu aşamadan sonra yapılacak işlem ise, modeli sanal düzlemler ile kesiştirerek dış hatlarını elde etmektir.

3.2 Model – Düzlem Kesişimi ile Dış Hatların Belirlenmesi

Sanal heykeltıraşlık sistemlerinde çok popüler olmasına karşın voxel modellemesinde sistem kaynaklarının aşırı tüketimi ve birim elemana dayalı olarak çok fazla hesaplama yapılması dexel modellemesinin ortaya atılmasına neden olmuştur. Ancak önerdiğimiz yöntem ile birlikte, dış hatların elde edilmesi esnasında, ışınlar yerine sanal düzlemler kullanılmaktadır. Sanal düzlemlerin kullanımı, daha az hesaplama işlemi kullanılarak nesne ile kesişimi daha hızlı gerçekleştirmektedir.

Sanal düzlemler, algoritmanın işlemesi gereken koordinat yükünü sadece dört koordinata indirger. Böylece görüntü listelerinde, gönderilecek ışınlar ve dexel noktalarının tutulması zorunluluğu da ortadan kalkmaktadır. Bu durum hem ekran kartının, hem işlemcinin, hem de hafızanın yükünü azaltacağı için sistemin performansını olumlu yönde etkilemektedir.

Bir düzlem ile üçgenin kesişimi bir doğru parçasıdır (Şekil 3.2.1) ilkesi kullanılarak düzlem ile aynı seviyede bulunan üçgenlerin sıralı kesişmelerinden düzlem üzerindeki dış hatlar modelin kalitesinden taviz vermeden elde edilir.



Şekil 3.2.1: Düzlem ile Üçgenin Kesişimi

Üçgen-Düzlem kesişmesi matematiksel olarak irdelendiğinde bu aşamada yapılan işlemler daha kolay anlaşılabilmektedir. Düzlemlerin birçok gösterimi olmasına karşın, genel gösterimleri şu şekildedir:

$$Ax + By + Cz + D = 0. \quad (3.2.1)$$

(1.1)'deki gösterime uyularak 2 düzlem şu şekilde tanımlanır:

$$A_1x + B_1y + C_1z + D_1 = 0, \quad (3.2.2)$$

$$A_2x + B_2y + C_2z + D_2 = 0, \quad (3.2.3)$$

Bu iki düzlemin kesişimi

$$(x-x_1)/a = (y-y_1)/b = (z-z_1)/c \tag{3.2.4}$$

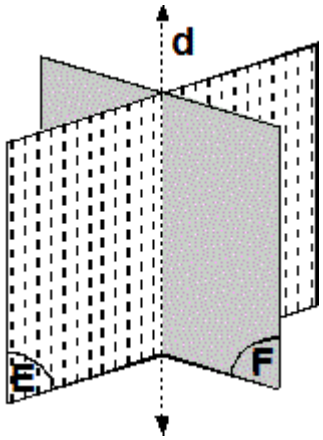
denklemi ile belirlenen doğrudur. (1.4)'te tanımlanan a, b, c, x_1, x_2, x_3 değerleri şu şekilde bulunur:

$$\left.\begin{aligned} a &= \begin{vmatrix} B_1 & C_1 \\ B_2 & C_2 \end{vmatrix}, \\ b &= \begin{vmatrix} C_1 & A_1 \\ C_2 & A_2 \end{vmatrix}, \\ c &= \begin{vmatrix} A_1 & B_1 \\ A_2 & B_2 \end{vmatrix}, \end{aligned}\right\} \tag{3.2.5}$$

$$\left.\begin{aligned} x_1 &= \frac{b\begin{vmatrix} D_1 & C_1 \\ D_2 & C_2 \end{vmatrix} - c\begin{vmatrix} D_1 & B_1 \\ D_2 & B_2 \end{vmatrix}}{a^2+b^2+c^2}, \\ y_1 &= \frac{c\begin{vmatrix} D_1 & A_1 \\ D_2 & A_2 \end{vmatrix} - a\begin{vmatrix} D_1 & C_1 \\ D_2 & C_2 \end{vmatrix}}{a^2+b^2+c^2}, \\ z_1 &= \frac{a\begin{vmatrix} D_1 & B_1 \\ D_2 & B_2 \end{vmatrix} - b\begin{vmatrix} D_1 & A_1 \\ D_2 & A_2 \end{vmatrix}}{a^2+b^2+c^2}, \end{aligned}\right\} \tag{3.2.6}$$

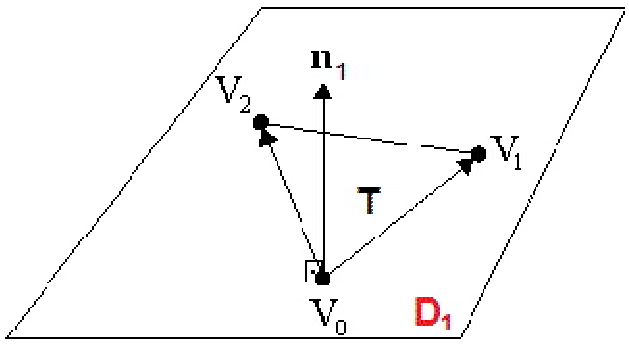
Eğer, a = b = c = 0 eşitliği söz konusu ise bu düzlemler paraleldir.

Şekil 3.2.2'de görülen E düzleminin ışın demetleri ile oluşturulduğu varsayılırsa, sanal ortamda modellenmiş F düzlemi ile kesişmesinden elde edilen d doğrusunun, ulaşılmak istenen dış hat dilimi olduğu görülebilir.



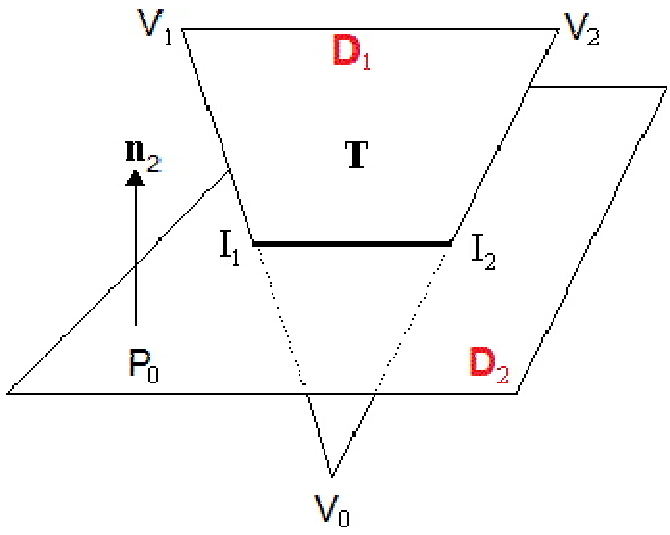
Şekil 3.2.2: Paralel Olmayan İki Düzlemin Kesişimi

Uygulama üçgenler (poligon) ile düzlemlerin kesişimi esasına dayalı olduğundan, düzlem-düzlem kesişimi, düzlemlerden herhangi bir tanesi boyunca bir üçgen uzandığı düşünülerek genellenebilir (Şekil 3.2.3).



Şekil 3.2.3: D_1 Düzlemi Üzerindeki T üçgeni

Böylece iki düzlemin kesişimi için tanımlanmış olan yapı düzlem-poligon kesişimi için de genellenmiş olacaktır. Şekil 3.2.4'te D_1 düzlemin boyunca uzanan ve V_0, V_1, V_2 köşe noktaları ile tanımlanan bir T üçgeni ile D_2 düzleminin kesişimi gösterilmiştir.

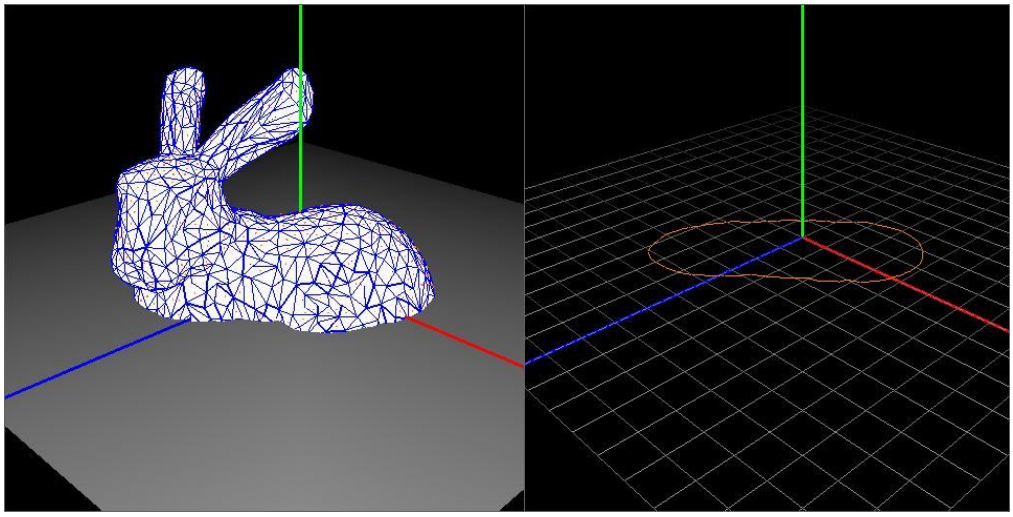


Şekil 3.2.4: T üçgeni ile D_2 Düzleminin Kesişimi

T üçgeni ile D_2 düzlemi ancak birbirlerine paralel ise kesişmezler. Eğer kesişiyorlarsa üçgenin tepe noktalarından bir tanesi D_2 düzlemi ile aynı tarafta olmak zorundadır. Aynı tarafta olan bu tepe noktası diğer iki tepe noktası ile segment oluşturur ve kesişim bu iki segment arasındaki doğru boyunca uzanır.

Şekil 3.2.4'te görüldüğü üzere V_0 noktasının D_2 düzlemi ile aynı tarafta kaldığını varsayılsın. Bu durumda T üçgeni, V_0V_1 ve V_0V_2 segmentleri I_1 ve I_2 noktalarında D_2 düzlemi ile kesişir. I_1I_2 doğrusu, T üçgeni ile D_2 düzleminin kesişim doğrusudur.

Sistemde, üç boyutlu nesne üzerine sadece tek ya da birbirine paralel bir dizi sanal düzlem gönderilebilir. Sadece bir adet düzlem gönderilmesi durumunda, tek düzlem üzerinden dış hat elde edileceğinden, sadece düzlem-model kesişmesinin olduğu alanlara sahip dış hat ortaya çıkacaktır (Şekil 3.2.5).

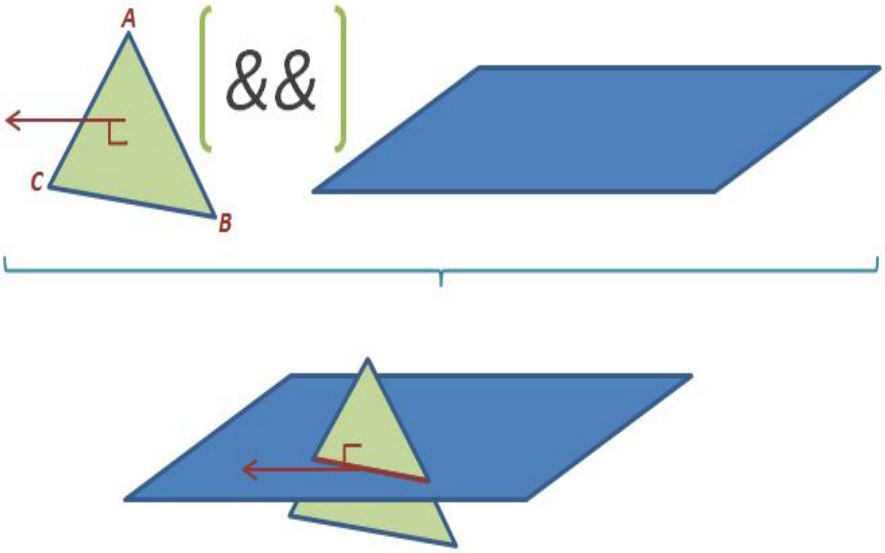


Şekil 3.2.5: Dış Hat Belirleme Süreci

3.3 Dış Hatların Normal Vektörlerinin Belirlenmesi

Dış hattı belirten poligonun üzerindeki her doğru parçası bir üçgen ile düzlemin algoritma üzerinden kesişim testine girmesiyle elde edilmiştir. Modeli oluşturan bu üçgenlerin normalleri ise düzlemle kesişmesinden oluşturdukları doğru parçalarına nakil edilir.

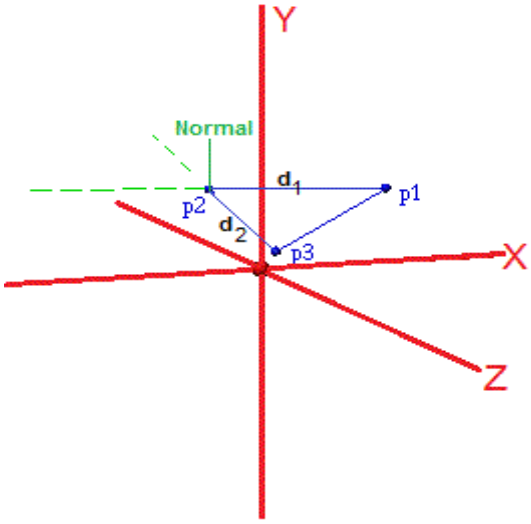
Şekil 3.3.1 Sanal heykeltıraşlıkta normal vektörleri ile ait oldukları objeler için önemli birer alt birimdir. Önerdiğimiz sistemde de daha sonra ki aşamalarında yapılacak hesaplamalar, kıstaslar ve optimizasyon işlemleri için normal vektörleri oldukça önemlidir.



Şekil 3.3.1: Üçgen ve Doğru Parçasının Normal Vektörü

Bilgisayarlı grafik için normal vektörlerinin elde edilmesi, hem ışıklandırma hem de kesişim testleri için önemli olduğu kadar dallanma aşaması için de önemi büyüktür. Çünkü bu aşamada normal vektörleri ve onlara dik olan vektörler algoritmadaki hesaplamalar için hayati kıstaslar oluşturmaktadırlar. İşlemlerin daha hızlı yapılabilmesi ve normal vektör hesaplamalarının tekrarlanmaması için çalışma ortamına aktarılan modeli oluşturan üçgenlerin normal vektörleri hesaplanarak, gerektiğinde derlenmek üzere görüntü listelerinde saklanırlar. Böylece işleyici alet ile etkileşime giren nesnenin modellenmesi sürecinde hesaplamalar çok daha hızlı şekilde yapılabilmektedir.

Uygulama üzerinde normal hesaplamasının nasıl yapıldığını anlamak için Şekil 3.3.2 üzerinde görülen durum referans alınırsa, ilk yapılması gereken, p_1, p_2 ve p_3 köşe nokta koordinatlarının kullanılarak, belirlenecek 2 vektör ile, bu üçgenin üzerinde bulunduğu yüzeyin düzleminin bulunmasıdır.



Şekil 3.3.2: Üçgenin Normalinin Bulunması

Üçgenin köşe noktaları, $p_1=(x_1, y_1, z_1)$, $p_2=(x_2, y_2, z_2)$, $p_3=(x_3, y_3, z_3)$ şeklinde tanımlanmış olsun. Üçgen üzerinde p_2 noktası referans olacak şekilde elde edilecek 2 vektör şunlardır:

$$d_1 = (x_2 - x_1,\ y_2 - y_1,\ z_2 - z_1) \quad (3.3.1)$$

$$d_2 = (x_3 - x_2,\ y_3 - y_2,\ z_3 - z_2) \quad (3.3.2)$$

Vektörler tanımlandıktan sonra her iki vektöre dik olan vektör bulunur. Bu sebeple, bu iki vektörün, vektörel çarpımının elde edilmesi gerekmektedir.

$$V[x] = d_1[y] * d_2[z] - d_1[z] * d_2[y] \quad (3.3.3)$$

$$V[y] = d_1[z] * d_2[x] - d_1[x] * d_2[z] \quad (3.3.4)$$

$$V[z] = d_1[x] * d_2[y] - d_1[y] * d_2[x] \quad (3.3.5)$$

Son olarak yapılacak işlem, normal vektörünün normalize edilmesidir:

$$length = \sqrt{V[x]^2+V[y]^2+V[z]^2} \quad (3.3.6)$$

Normalizasyonu sağlayacak length değeri de hesaplandıktan sonra, her iki vektöre dik olan vektörün x, y ve z koordinatları bu değere bölünerek normal vektörü elde edilir.

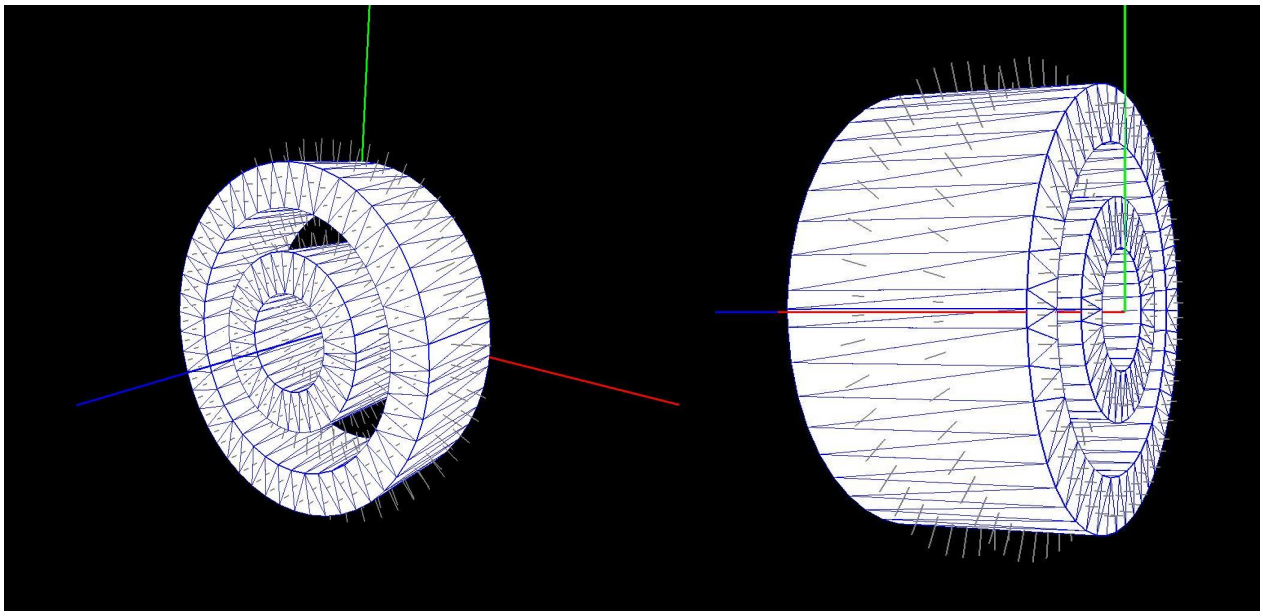
$$NORMAL[x] = V[x] / length \quad (3.3.7)$$

$$NORMAL[y] = V[y] / length \quad (3.3.8)$$

$$NORMAL[z] = V[z] / length \quad (3.3.9)$$

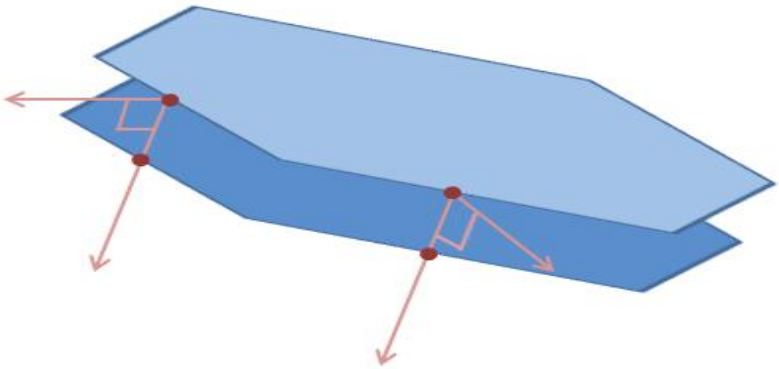
Hesaplanan normal vektörü, referans alınan köşe noktası üzerinden çizilmektedir. Ancak Şekil 4.3.2'ye bakıldığında normallerin, üçgenlerin ağırlık merkezinden çizildiği görülmektedir. Bu işlem için ayrıca üçgenlerin koordinatlarına göre özel bir hesaplama yapılarak normal vektörünün başlangıç noktası, referans köşe noktasından üçgenin ağırlık merkezine ötelenir.

Tüm bu hesaplamalardan sonra modeli oluşturan üçgenlerin normalleri elde edilir. Şekil 3.3.3'te farklı açılardan "çemberler(circles)" isimli modeli oluşturan üçgenlerin normalleri görülmektedir.



Şekil 3.3.3: Normallerin Model Üzerinde Gösterimi

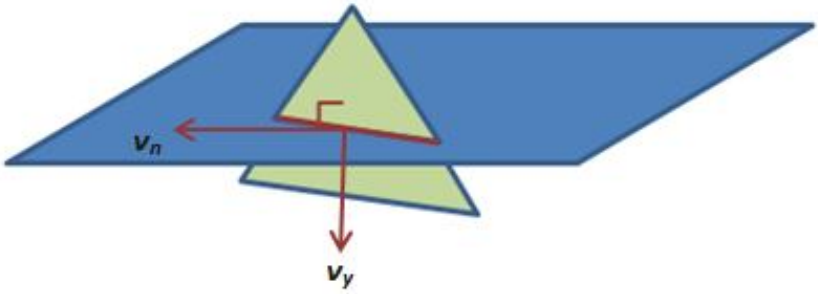
Normaller, hesaplandıktan sonra, üçgenleri oluşturan köşe noktaları ile birlikte gerektiğinde tekrar tekrar kullanılmak üzere görüntü listesinde kaydedilirler. Sanal düzlemler ile kesiştirilen model daha sonra dış çeper katmanları ile tanımlanır ve elde edilen dış çeperlerde normal ve normale dik vektörler tutulur. Bu vektörlerin yönleri dış hatların birbirleri ile doğru eşleşmesinde önemli bir kıstas olacaktır.



Şekil 3.3.4: Dış Hatlar Üzerinde Tutulan Normal ve Normale Dik Vektörler

3.4 Dış Hatların Yön Vektörlerinin Belirlenmesi

Dış hattı oluşturan poligonların birim elemanları olan doğru parçaları kullanılan veri yapısı içerisinde normal vektörler ile beraber saklanmaktadır. Bu normal vektörleri, doğru parçasının vektörel ifadesi ile vektörel çarpımı sonucunda yön vektörleri bulunur. Kullanılan veri yapısı içerisinde doğru parçası ve normal vektörü ile beraber saklanır (Şekil 3.4.1).



Şekil 3.4.1: Yön Vektörünün Gösterimi

Önerilen sistemde kilit rol oynayan bu yön vektörleri, düzlemler üzerindeki paralel dış hatların tekrar üçgenleştirilmesinde kullanılacaktır.

3.5 Dış Hatların Sıralanması

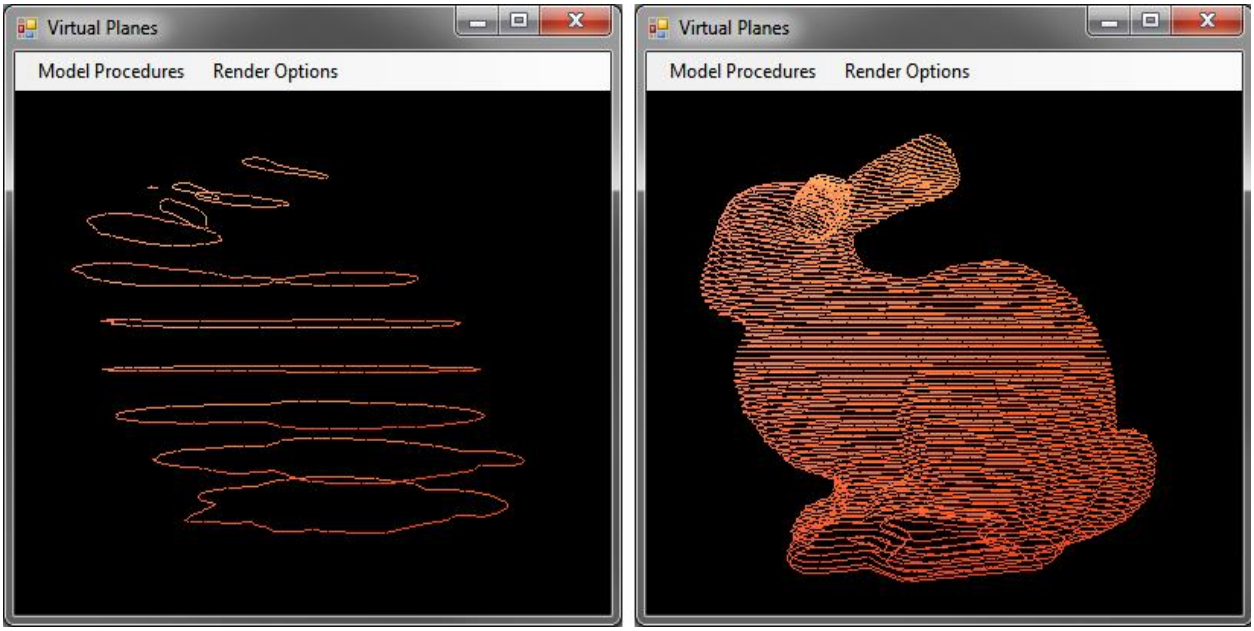
Rastgele sıralanan üçgenlerin düzlem üzerinde rastgele kesişmesi sonucu doğru parçaları gelişigüzel yerleşecektir. Bu gelişi güzel sıralanma dış hatların belirlenmesinde yeterli olsa da, dış hatların kesici aleti temsil eden poligonlar ile etkileşimi sırasında yapılacak hesaplamalar için yeterli olmayacaktır.

Kesişim ve birleşim işlemleri için dış hattı oluşturan doğru parçalarının saat yönünde ya da saat yönünün tersine doğru sıralanması gerekmektedir.

3.6 Dış Hatlar ile Modelin Gösterimi

Eğer tek bir düzlem değil de birbirine paralel bir dizi sanal düzlem model üzerine gönderilirse, nesne dış hat çeperleriyle yeniden inşa edilmiş olur. Ancak sanal düzlemlerin sıklığı nesnenin detaylı bir şekilde inşası için önemli bir parametredir.

Gönderilen düzlemlerin sıklığı, nesnenin çözünürlüğüyle doğru orantılı olduğundan, sıklık arttıkça görünüm daha gerçekçi ve keskin olacaktır. Şekil 3.6.1 de sanal düzlem sıklığının netlik açısından önemi görülmektedir.



Şekil 3.6.1: Sanal Düzlem Sıklığı

Modelin çok fazla sayıda dış hatla gösterimi her zaman olumlu bir durum olmayabilir. Dış çeper sayısı arttıkça algoritma üzerinde yapılacak kesişim testleri ve hesaplamalarda artacağından sistem üzerine fazla yük binecektir. Modeli az sayıda dış çeper ile tanımlamak ise yetersiz modelleme ve tatmin etmekten uzak sonuçlar doğuracağından dış hat sayısı için uygun bir değer belirlemek gerekmektedir.

Modelin dış çeperler ile gösterimine kadar geçen aşamalar, işleme kısmı için bir hazırlık olarak görülebilir. Bundan sonraki bölümde, işleyici alet ile modelin etkileşimi ve bu etkileşime dayalı olarak modelin yeniden inşası anlatılacaktır.

4. ÖNERİLEN METHOD

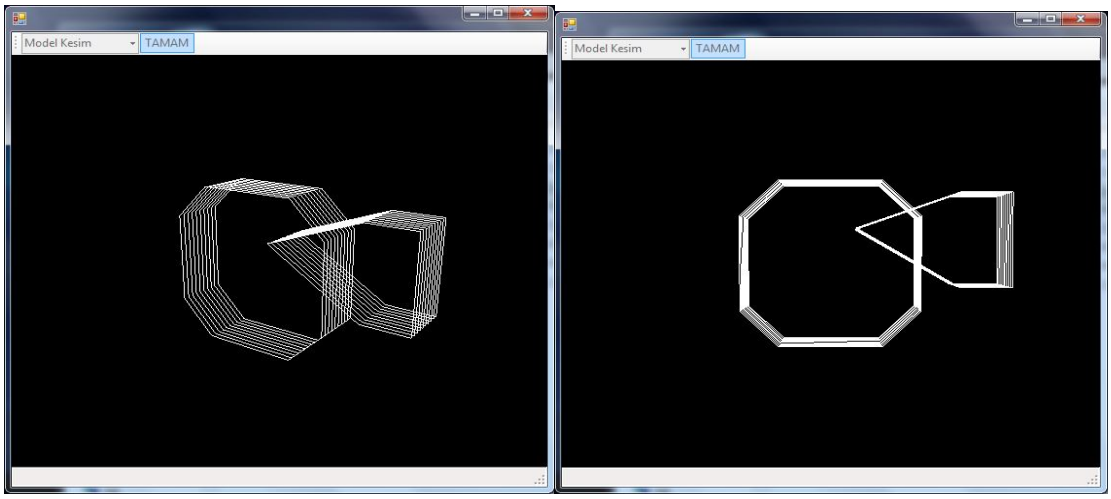
Dördüncü bölümde bu tez çalışmasının kaynağı olan Sanal Düzleme dayalı yeniden inşa metodu üzerine kurulmuştur. Önerilen metodun çalışma yapısı, diğer çözümlerden farkı, birtakım aykırılıklara karşı verdiği sonuçlar, ayrıntılı olarak resimlerle anlatılmıştır.

Modelin yeniden inşası, model-alet etkileşiminden katı görünüme geçişe kadar olan süreç boyunca detaylandırılmıştır.

4.1 Model – Alet Etkileşimi

İşleyici aletin model üzerinde etkileşime gireceği konum ayarlanır ve istenen yere gelindiğinde etkileşim için gerekli hesaplamalar yapılır. Ancak bu hesaplamalar iteratif olarak yapılarak tıpkı gerçek hayattaki heykeltıraşlık işlemlerine benzer şekilde üç boyutlu sanal ortama yansıtılması hedeflenmektedir.

Şekil 4.1.1'de görüldüğü üzere dış hatları belirlenen işleyici alet ve işlenecek model birbirleri ile etkileşime girmektedir. Bu esnada program yön kontrolleri işleyici aleti hareket ettirmektedir.



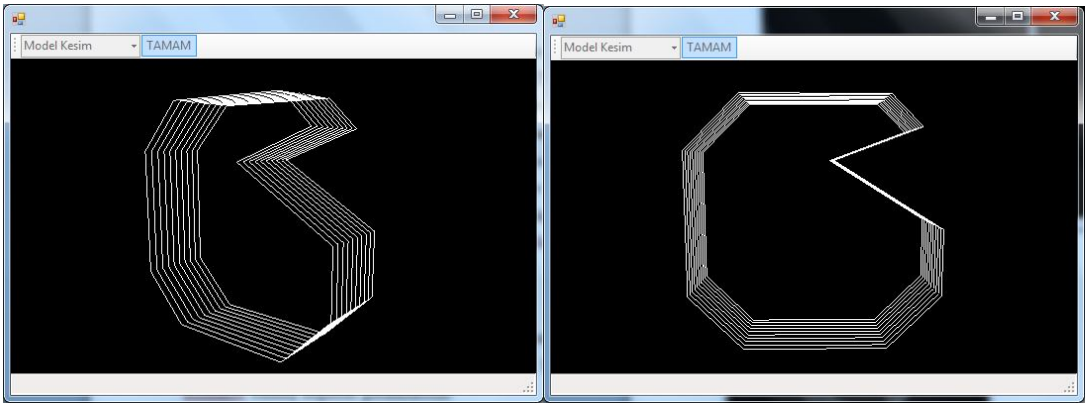
Şekil 4 1.1: İşleyici Alet ve Model Etkileşiminin Gösterimi

4.2 Model–İşleyici Alet İşlemleri

Bu aşamada model üzerinde istenen konuma gelindikten sonra iki tür davranış söz konusudur. İlk davranış türü, kesişimin model üzerinden çıkarılması; kesme(koparma) işlemidir. Bu işlem kesişim(intersection) fonksiyonu üzerine tanımlanmıştır.

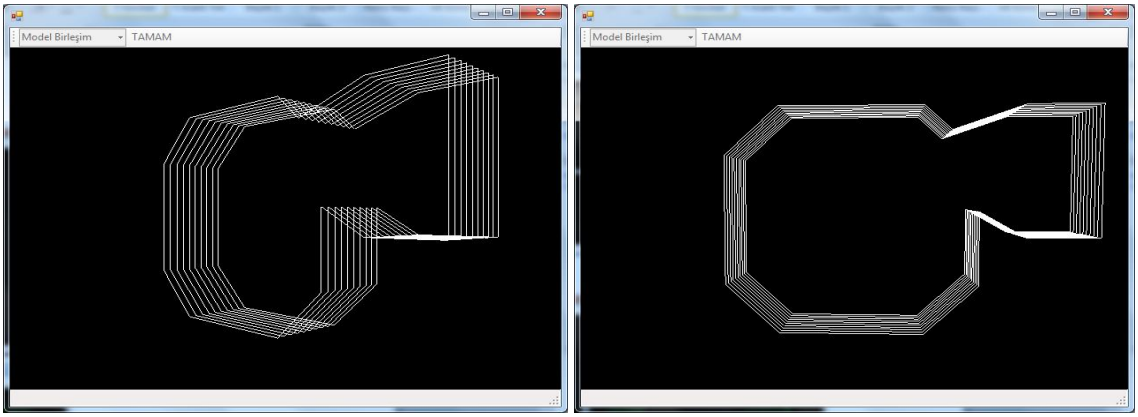
İşleyici aletin model ile kesiştiği alan, modelden çıkarılması istenen yerler olarak algılanır ve hesaplamalar buna göre yapılır. Bu hesaplamalarda dikkat edilmesi ve göz önünde bulundurulması gereken en önemli detay, işleme sırasında işleyici aletin geometrik yapısının işlenen model üzerine de doğru bir şekilde yansıtılmasıdır. Dolayısıyla oval hatlara sahip bir işleyici aletin, model üzerindeki teması da benzer şekilde oval hatlar oluşturmalıdır.

Şekil 4.2.1'e bakıldığında Şekil 4.1.1'de görülen işleyici alet-model kesişimi sonrası yapılan hesaplamalar sonucu, modelin son hali görülmektedir.



Şekil 4.2.1: Kesişim Sonrası Modelin Son Hali

Bu aşamadaki bir diğer davranış türü ise eklemedir. Ekleme ile birlikte istenen konuma getirilen işleyici alet, model üzerine birleşim(union) fonksiyonunu kullanarak eklemeler yapmaktadır. Şekil 4.2.2'de, Şekil 4.2.1'deki işleyici alet-model etkileşiminin ekleme fonksiyonuna göre aldığı durum gösterilmektedir.



Şekil 4.2.2: Model Üzerine Ekleme Yapılması

4.3 Kesişim(Intersection) ve Birleşim(Union) Algoritması

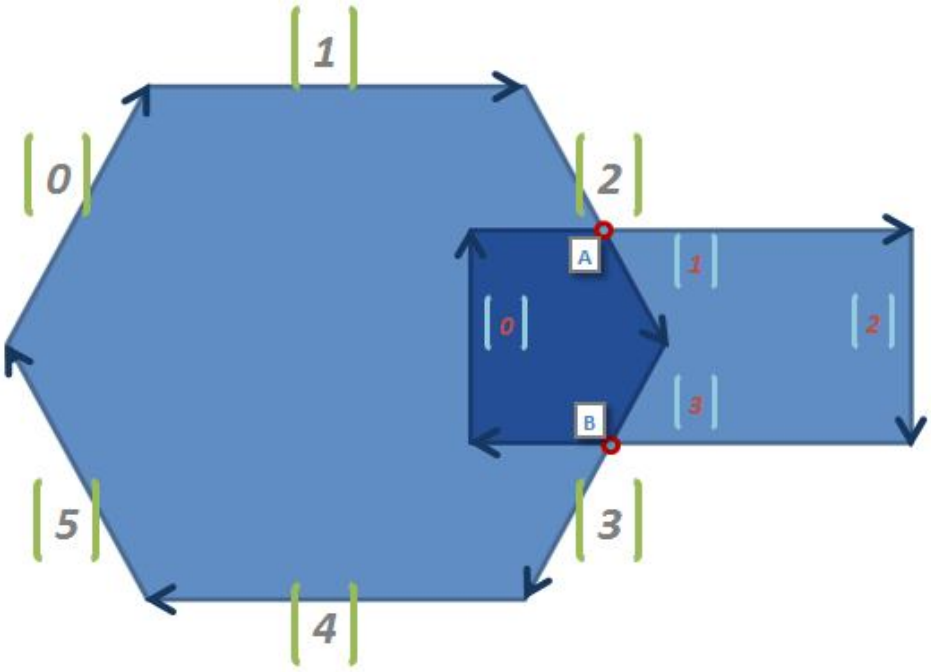
Bu tez çalışması için hazırlanan kesişim ve birleşim algoritmasında, poligonları oluşturan doğru parçaları saat yönünde sıralanarak bağlı listelerde tutulmaktadırlar. Algoritmanın yapısı, bu bölümde yazılan pseudo kod ile daha ayrıntılı bir şekilde ve adım adım anlatılmıştır.

Burada dikkat edilmesi gereken özelliklerden bir tanesi de poligonların doğru parçalarından oluşmasıdır. Bu doğru parçaları modelin sanal düzlemler ile kesişimleri sonucu elde edilir. Sıfırıncı indeks ataması ilk kesişen üçgene göre belirlenir.

1. Üçgenlerden oluşan modelin ve sanal düzlemin kesişimi ile elde edilen poligonlar üzerinde, ilk kesişen doğru parçasının indeksine 0 atayarak saat yönünde sıralayıp indeks belirle.
2. Poligonlar saat yönünde sıralandığı için her doğru parçasının ilk noktasını "A" ikinci noktasını "B" olarak kabul et.
3. Modeli oluşturan poligon ile kesici aleti oluşturan poligonun kesim noktalarını ve kesim indekslerini bul.
4. Poligonun sıfırıncı indeksinden başlayarak ilk kesişen doğru parçasına kadar olan bütün doğru parçalarını yeni oluşan poligona ekle.
5. Kesişim algoritmasında, kesim noktalarının, modeli oluşturan maksimum ve minimum X-ekseni değerlerine göre kesici aletin dış hat çizgilerini saat yönünün tersine ya da saat ekseni yönünde ekle.
6. İkinci kesim noktasına kadar kesici aletin dış hat çizgilerini sonuç poligonuna ekle.
7. İkinci kesim noktasından sonra modelin dış hat çizgilerini eklenmeye devam et ve bu işlemi son indekse ulaşana kadar sürdür.

Yukarıdaki adımların sözde kod(Pseudocode) olarak şu şekilde verilebilir:

```
VirtualPlane vp = VirtualPlane[idx];

        foreach(triangle t in model)
        {
        LineArray.Add(intersect(t, vp));
        }

LineArray.Sort((IComparable)CW);

        foreach(polygon p1 in vp_model[idx].Polygons)
        {
                foreach(polygon p2 in vp_digger[idx].Polygons)
                {
                 IntersectionArray.Add(Intersect(p1, p2));
                }
        }

for(int i=0; i<IntersectionArray[0]; i++)

{
        ResultArray.Add(vp_model[idx].Polygons.Lines[i]);
}

for(int i=IntersectionArray[0]; i< IntersectionArray[1]; i++)

{
   if(ClipType == Intersection)
   {
    IndexDiggerIntersection=TestPointInTheBox(vp_digger[idx].Polygons.Lines[i]);
    ResultArray.Add(vp_digger[idx].Polygons.Lines[IndexDiggerIntersection]);
   }

  if(clipType == Union)

  {
  IndexDiggerUnion = TestPointInTheBox(vp_digger[idx].Polygons.Lines[i]);
  ResultArray.Add(vp_digger[idx].Polygons.Lines[IndexDiggerUnion]);
   }
}

for(int i= IntersectionArray[1]; i< vp_model[idx].Polygons.Lines[i]); i++)

{
        ResultArray.Add(vp_model[idx].Polygons.Lines[i]);
}
```

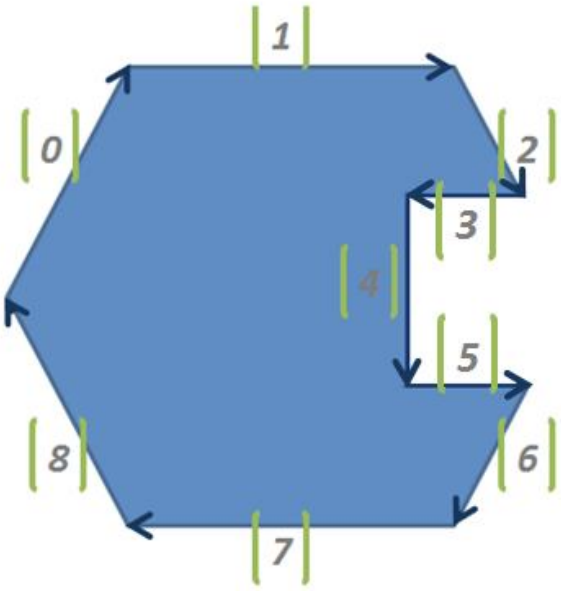


Şekil 4.3.1: Kesişim ve Birleşim Algoritmaları için İndeks Yapısı

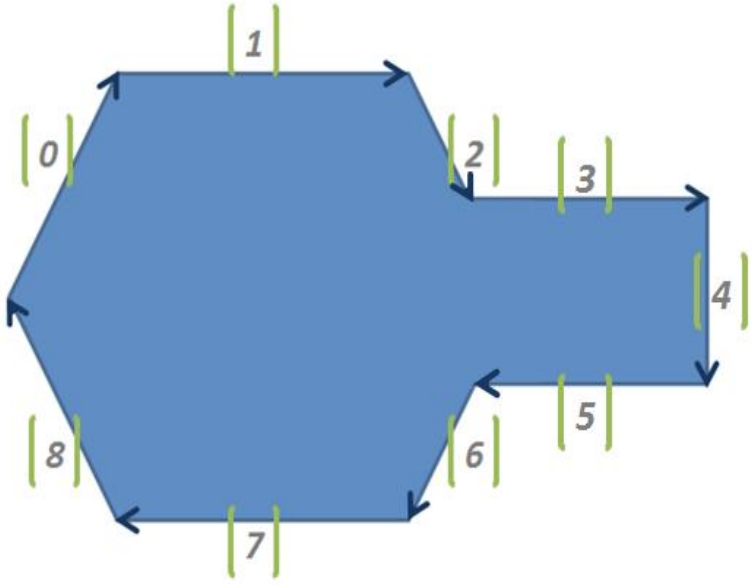
Şekil 4.3.1 üzerinde hem model hem de işleyici alet üzerine yapılan indekslemeler gösterilmiştir. Şekilde görüldüğü üzere 6 indeksli model ile 4 indeksli kesici alet etkileşime girmiştir.

0 (Sıfır) başlangıç indeksleri, modelin sanal düzlemler ile kesişimine bağlı olarak belirlenir. İlk kesişen poligona sıfır indeksi verilmektedir. Daha sonra saat yönünde bir yol takip edilerek indeks değerleri arttırılır. Şekil 4.3.1'de görülen A ve B noktaları "kesişim noktaları" olarak tanımlanmaktadır. A ve B noktalarından sonra takip edilecek yol, kesişim veya birleşim fonksiyonuna göre şekillenmektedir. Şekil 4.3.2'de kesişim işlemi sonucu modelin aldığı son durum gösterilirken, Şekil 4.3.3'de ise modelin birleşim fonksiyonuna tabi tutulduktan sonra ki durumu gösterilmektedir.

Her iki resimden de anlaşılacağı üzere kesişim ve birleşim işlemleri modelin indeks sayılarını değiştirmektedir. Dolayısıyla iteratif olarak tekrarlanan bu işlemlere bağlı olarak sürekli yinelenen kontroller söz konusudur. Bu kontrollerin optimizasyonu gelecekte yapılması planlanan bir çalışmadır.



Şekil 4.3.2: Kesişim İşlemleri Sonucu Modelin Yeni Görünümü



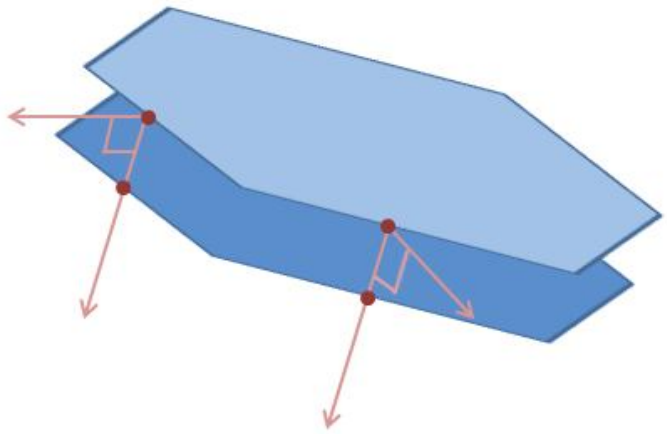
Şekil 4.3.3: Birleşim İşlemleri Sonucu Modelin Yeni Görünümü

4.4 Dış Hatlar Arası Üçgenleştirme İşlemi (Triangulation)

Önerilen metodun en önemli aşamasıdır. Bu aşamada model tamamen sanal düzlemlerle temsil edilmektedir. Üçgenleştirme işlemi ile birlikte sanal düzlemler arasında kalan alanlar birleştirilerek modelin, sanal heykeltıraşlık işlemleri sonrası katı görünümüne bir adım daha yaklaşması sağlanır.

İki sanal düzlem arasında üçgenleştirmenin nasıl yapılacağına karar verilecek aşama burasıdır. Her düzlem üzerindeki dış hatların veri yapılarında, poligonu oluşturan doğru parçaları, normal vektörleri ve yön vektörlerinin saklandığı daha önceki bölümlerde belirtilmişti.

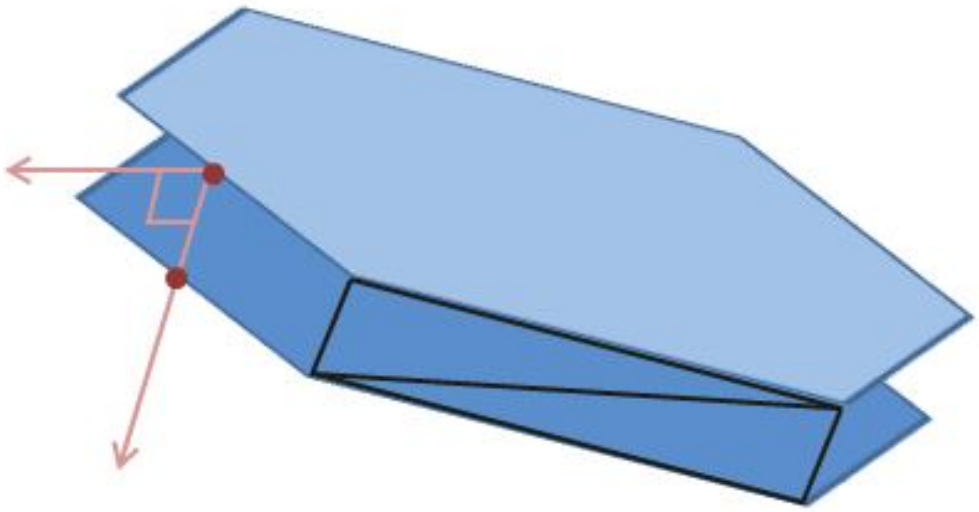
Kesişim ve birleşim işlemlerinden sonra yeniden oluşturulan dış hatların birbirleri ile yeniden üçgenleştirilmesi modelin katı hale gelmesini sağlayacaktır (Şekil 4.4.1).



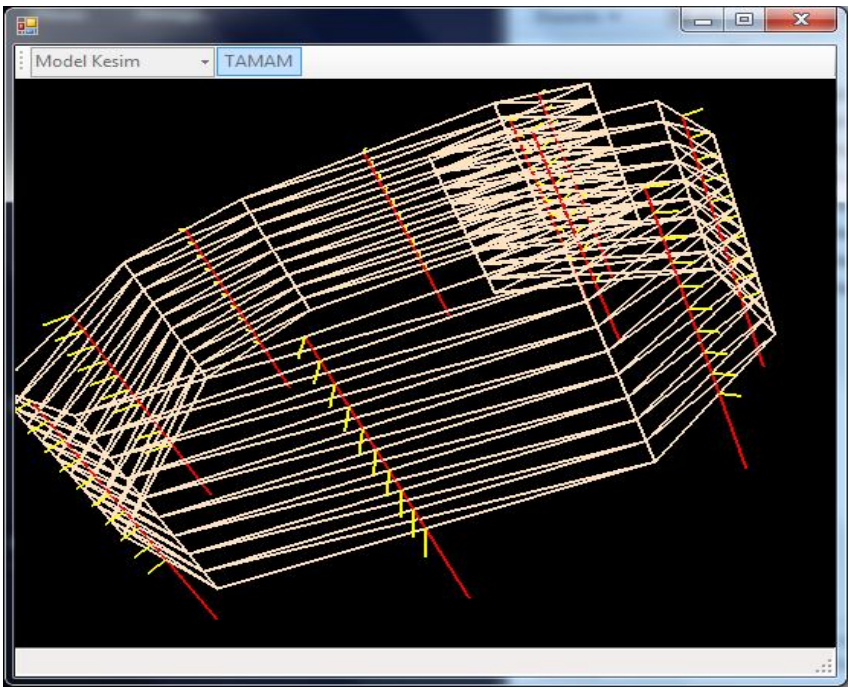
Şekil 4.4.1: Sanal Düzlemlerin Birbirine Göre Durumu

Önerilen sistemde iki sanal düzlemin birbirleriyle yeniden üçgenleştirilmesi yön vektörlerinin kullanılmasıyla sağlanmaktadır.

Üst katmanda bulunan poligonların yön vektörleri alt katmandaki doğru parçası ile kesişirse, üst katmandaki doğru parçası ile alt katmandaki doğru parçası arasında yeniden üçgenleştirme yapılır (Şekil 4.4.2 ve Şekil 4.4.3).

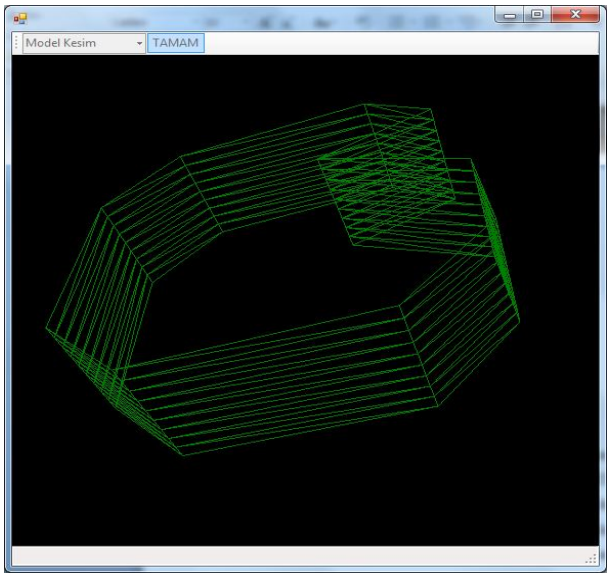


Şekil 4.4.2: Sanal Düzlemler Arasında Üçgenleştirme



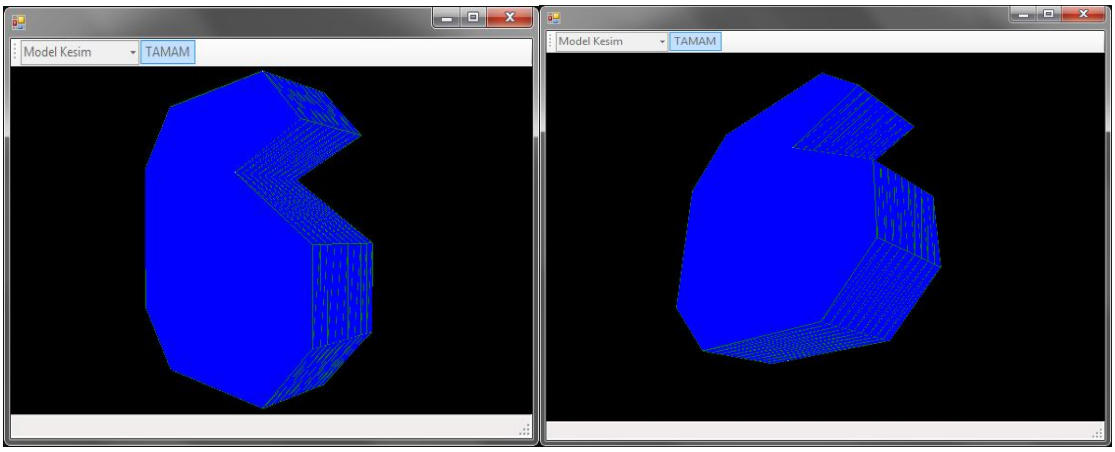
Şekil 4.4.3: Sanal Düzlemler Arasında Üçgenleştirme

Sanal düzlemler arasındaki bu üçgenleştirme iteratif bir şekilde gerçekleştiğinde modelin yeniden üçgenlerle inşası sağlanmış olur (Şekil 4.4.4).



Şekil 4.4.4: Üçgenlerle Örülmüş Dış Çeperler

Normaller ve normale dik doğrular dikkate alınarak algoritma üzerinde yapılan hesaplamalar sonucu, dış hatlar arasında oluşturulan üçgen çeperler, bir süre sonra modelin katı görünümüne geçişi için kullanılacaktır. Katı görünüme geçirilen nesne modellemesi eğri algoritmaları kullanılarak daha keskin bir görünüme getirilebilir. Şekil 4.4.5'te dış çeperleri üçgenlerle örülmüş modelin katı(solid) görünüme geçişi gösterilmiştir.



Şekil 4.4.5: Modelin Katı(Solid) Görünüme Geçişi

5. PERFORMANS VE KARŞILAŞTIRMA

Bu bölümde belirtilen tüm performans ölçütleri Intel Pentium IV 3.40 GHz işlemci teknolojisine, 2GB fiziksel hafızaya, Windows 7 32-Bit işletim sistemine sahip, ekran kartı üzerindeki işlemci teknolojisi NVIDIA GeForce 7300LE olan, tek işlemci gücüne sahip bir bilgisayar üzerinde yapılmıştır.

5.1 Önerilen Sistemin Dış Hat Belirleme Performansı

Geliştirilen yeni dış hat yaklaşımı çerçevesinde performans iki ana başlık altında incelenebilir. Birincisi, modelin çalışma ortamına aktarılma süreci, ikincisi ise dış hat verilerinin sanal düzlemlerle kesişimi sonucu elde edilmesini kapsayan, dış hat hesaplama performansıdır.

Modelin çalışma ortamına aktarımında yapılanlar, bir metin veya raw dosyasından okunan koordinatların görüntü listelerine aktarılması ve aktarılan bu koordinatlar doğrultusunda üç boyutlu modelin ekrana yansıtılması sürecidir. Bu süreci kapsayan işlemin performansı, zaman kriterine göre değerlendirilir.

Performans değerlendirilmesi için öncelikle, tavşan(bunny), elma(apple), düğüm(knot), boğa(bull) ve iskelet(skull) olmak üzere 5 tane birbirinden farklı poligon sayısına, görünüme ve hacimsel yapıya sahip model seçilmiştir.

Seçilen modellerin koordinatları ilk olarak sistem dâhilinde geliştirilen DataChanger adlı program ile normalize edilmektedir. Normalize işlemi sonunda elde edilen yeni koordinat dosyası, eski dosyanın metin veya raw dosyası olmasına bakılmaksızın, daha düşük boyutlu bir metin dosyasına dönüştürülür ve dosya isminin sonuna "_2" son eki eklenir (Tablo 5.1.1).

Tablo 5.1.1: DataChanger ile Normalize Edilmiş Dosyaların Boyutları

	Dosya Boyutu	**Yeni Dosya Boyutu**	
apple.txt	13,40 MB	9,33 MB	apple_2.txt
bull.txt	1.91 MB	1,10 MB	bull_2.txt
bunny.txt	487 KB	280 KB	bunny_2.txt
knot.txt	259 KB	147 KB	knot_2.txt
skull.raw	72 KB	55 KB	skull_2.txt

Bir sonraki adımda nesnelerin çalışma ortamına aktarılma süreci değerlendirilmektedir (Tablo 5.1.2). Burada en önemli etken nesneyi oluşturan poligon sayısıdır. Poligon sayısı ile nesnenin ortama aktarılması için gerekli süre arasında doğru orantı vardır. Model ne kadar fazla poligon(üçgen) içeriyorsa aktarılma süreci de o kadar uzayacaktır. Ancak sistem içinde görüntü listeleri kullanıldığından ilk yüklemelerde daha fazla zaman geçmesi muhtemel bir durumdur. Fakat algoritma tarafından yapılacak ileri aşamalardaki kesişim testlerinde, bu veriler daha hızlı bir şekilde çalıştırılabilir.

Tablo 5.1.2: Modellerin Ortama Aktarılma Performansı

	Poligon Sayısı	**Geçen Süre (ms)**
apple_2.txt	85.457	15329
bull_2.txt	12.398	2461
bunny_2.txt	2.984	2035
knot_2.txt	1.600	717
skull_2.txt	684	171

Sistemin performans değerlendirmesinin ikinci aşaması dış hatların belirlenme süreleri üzerinedir. Ortama aktarılmış olan modellerin düzlemlerle kesişmeleri sonucu, dış hatlarının belirlenmesinde geçen süre hesaplanmıştır. Hesaplamalar, modelin 100 tane birbirine paralel sanal düzlem ile kesişmesine göre yapılmıştır (Tablo 5.1.3).

Tablo 5.1.3: Dış Hat Belirleme Süreci

	Poligon Sayısı	**Geçen Süre (ms)**
apple_2.txt	85.457	1022
bull_2.txt	12.398	394
bunny_2.txt	2.984	278
knot_2.txt	1.600	114
skull_2.txt	684	42

Sistemin çalışma performansı, dexel ve voxel hacim modellerine kıyaslanarak değerlendirilmiştir. Kıyaslanan dexel ve voxel modelleri için görüntü listeleri oluşturulmamış olmasına karşın önerilen sistemde bu yapı kullanılmıştır.

Her üç modelin çalışma yapısı birbirinden farklı olduğu için değerlendirme sürecinde kullanılan zaman kıstasını tüm modeller için aynı tutmak kolay değildir. Örneğin, önerilen sistemde tek bir sanal düzlemin kesişmesini sağlamak için, düzlemin 4 koordinatının verilmesi yeterli olmasına karşın, aynı yapıyı dexel modelinde kurmak için yatay düzlemde birçok ışının gönderilmesi gerekmektedir.

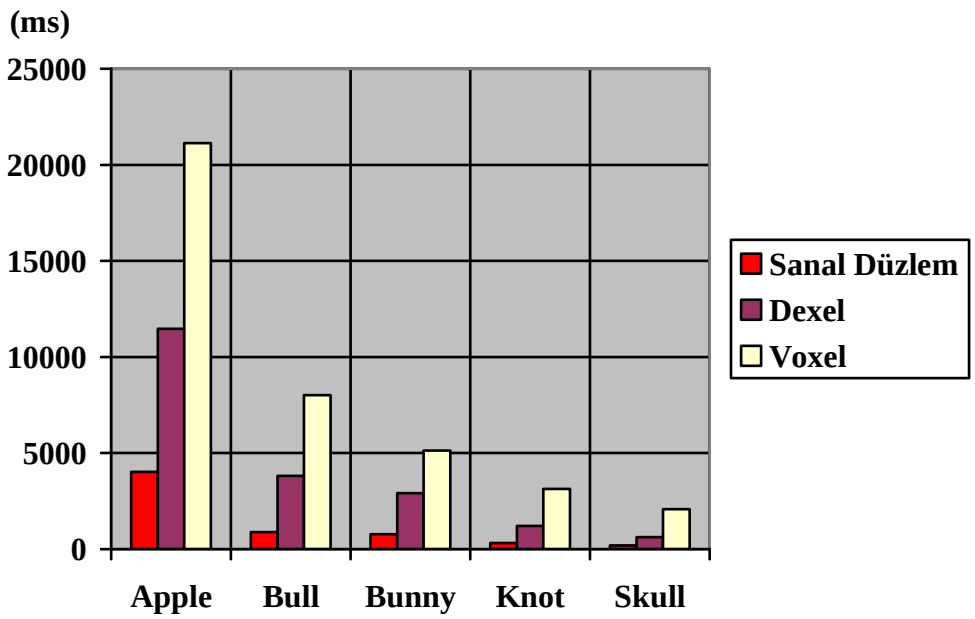
Tablo 5.1.4'deki performans değerleri, farklı modellemeler için nesne üzerine gönderilen birbirine paralel 100 katman için belirlenmiştir.

Tablo 5.1.4: Farklı Modellemeler için Dış Hat Belirleme Süreci

	Virtual Plane	**Dexel**	**Voxel**
apple_2.txt	4022 ms	11463 ms	21134 ms
bull_2.txt	894 ms	3816 ms	8014 ms
bunny_2.txt	778 ms	2914 ms	5129 ms
knot_2.txt	314 ms	1209 ms	3136 ms
skull_2.txt	196 ms	621 ms	2084 ms

Önerilen sistemde göze çarpan en büyük avantaj, dış hatlar belirlenirken sanal düzlemlerin işlenecek veriyi kendi köşe koordinatlarına indirgemesi olarak görülmektedir. Voxel modelindeki sayısız küp ile ışın kesişimi, dexel modelindeki bir yatay düzlem boyunca uzanan sayısız ışın ile nesneyi oluşturan üçgenlerin kesişimi yerini sadece bir düzlem ile nesnenin kesişmesine bırakmıştır.

Şekil 5.1'e bakıldığında önerdiğimiz modelin diğer örneklere kıyasla daha hızlı dış hat belirlediği görülmektedir. Bu durum hem görüntü listelerinde düzlem ve nesne koordinatlarının saklanması hem de düzlemin sadece 4 koordinatının tutularak model ile kesişime tabi olması ile açıklanabilir.



Şekil 5.1: Önerilen Sistemin Diğer Metotlarla Kıyaslanması

5.2 Önerilen Sistemin Yeniden İnşa Performansı

Yeniden inşa sürecinin performans değerlendirmesi, modelin işleyici alet ile etkileşime girmesi ve etkileşim sonrası yeniden inşa edilmesi olarak iki başlıkta incelenebilir. Her iki aşamada da İşlemci ve Fiziksel Bellek gibi sistem kaynaklarının ne derece tüketildiği önemli bir kıstastır.

Tablo 5.2.1'de 5 adet model için işleyici alet ile etkileşim sırasında kullanılan işlemci ve fiziksel bellek miktarı gösterilmektedir.

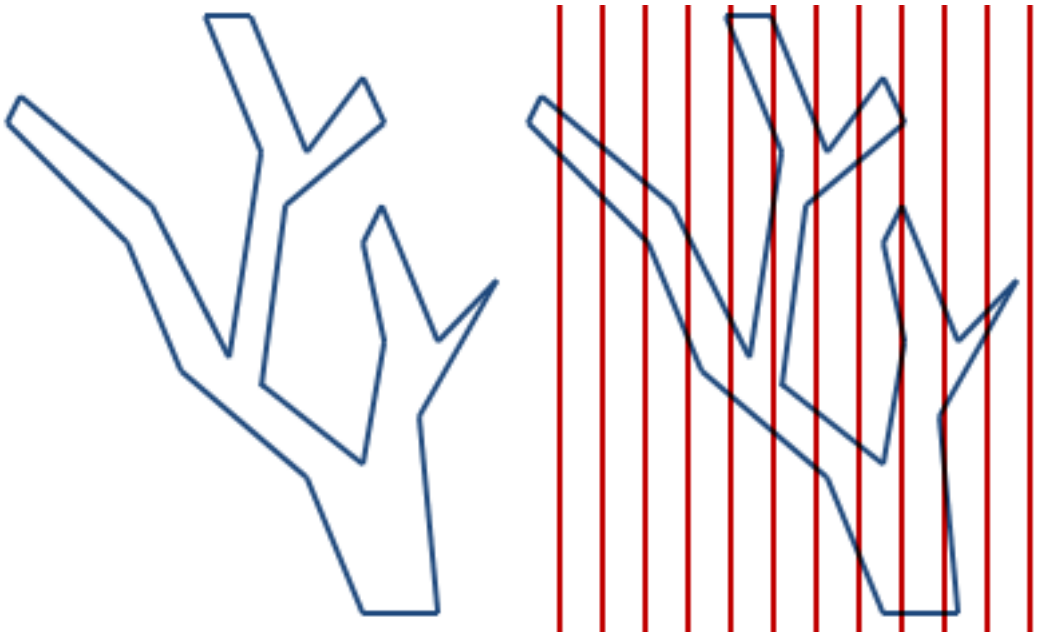
Tablo 5.2.1: Modellerin İşlenme Sürecinde Sistem Kaynaklarını Tüketme Performansı

	İşlemci (CPU)	**Fiziksel Bellek (MB)**
apple_2.txt	% 21	24 MB
bull_2.txt	% 9	11 MB
bunny_2.txt	% 4	4 MB
knot_2.txt	% 1	1 MB
skull_2.txt	% 1	0.4 MB

Görüldüğü üzere poligon sayısına bağlı olarak model üzerinde daha fazla dış çeper tanımlanacağından daha fazla işlemci gücü ve fiziksel bellek ihtiyacı duyacaktır. Ancak burada dikkat edilmesi gereken durum, sistem kaynaklarının işlenme süresi esnasında daha aşağı seviyelere çekileceğidir. Çünkü işleme sırasında sadece işleyici aleti kapsayan dış çeperler üzerinde işlem yapılır.

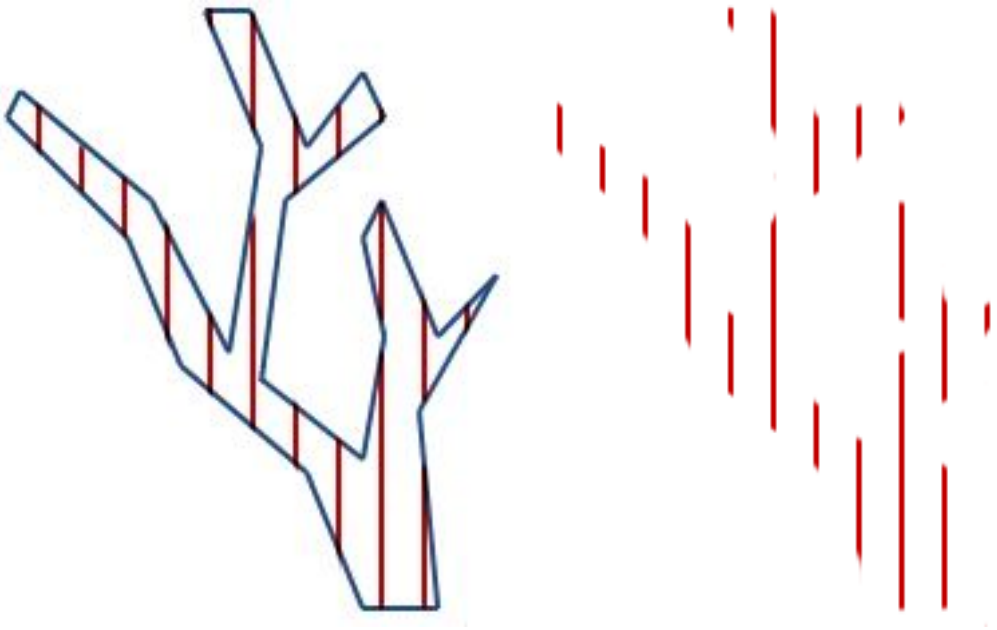
5.3 Anomaliler

İçbükey ve dış bükey yapıdaki modeller, algoritmalar üzerinde birtakım anomaliler meydana getirebilmektedir. Bu bölümde ortaya çıkabilecek anomalilerden bahsedilecektir. Şekil 5.3.1'e bakılırsa karmaşık bir model yapısı görülmektedir. Voxel modellemesine hızlı bir alternatif olarak ortaya çıkarılan Dexel modellemesinin bu model karşısında verebileceği hatalar ışın izleme algoritmasına dayalı olarak ortaya çıkar.



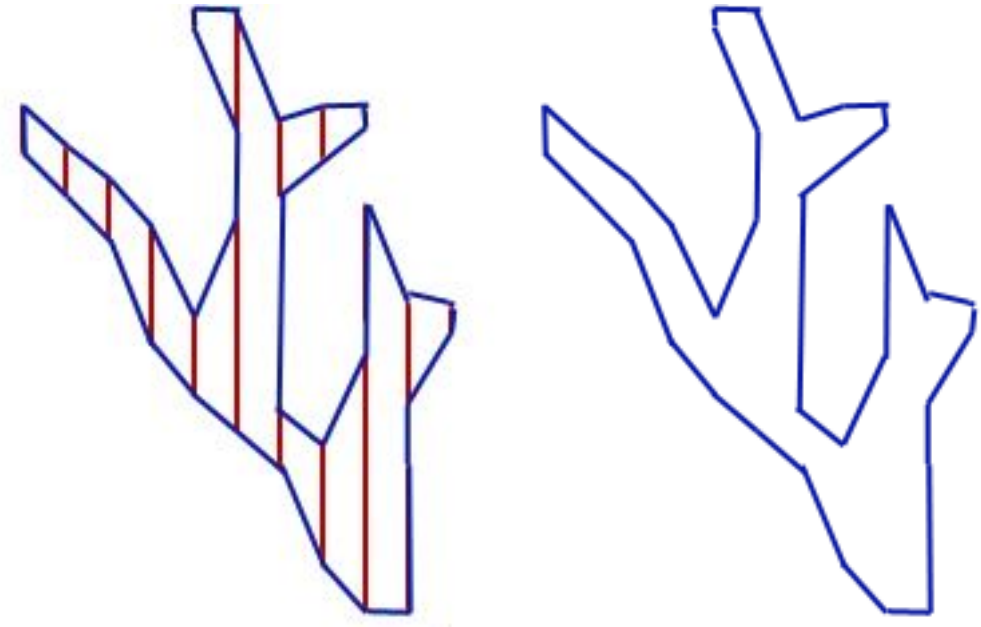
Şekil 5.3.1: Karmaşık Yapıdaki Bir Modelin Dexellerle Tanımlanması

Öncelikle model üzerine bir dizi ışın gönderilerek bu ışınlar modelin hacmini oluşturan dış yüzey ile kesiştirilir. Kesişime dayalı olarak ışınların nesne modelinin iç hacminde başka bir kesişim noktası olmadan takip ettiği kısımlar dexel olarak atanır ve modelin dexellerle ifadesi sağlanmış olur (Şekil 5.3.2).



Şekil 5.3.2: Objenin Dexeller ile İfade Edilmesi

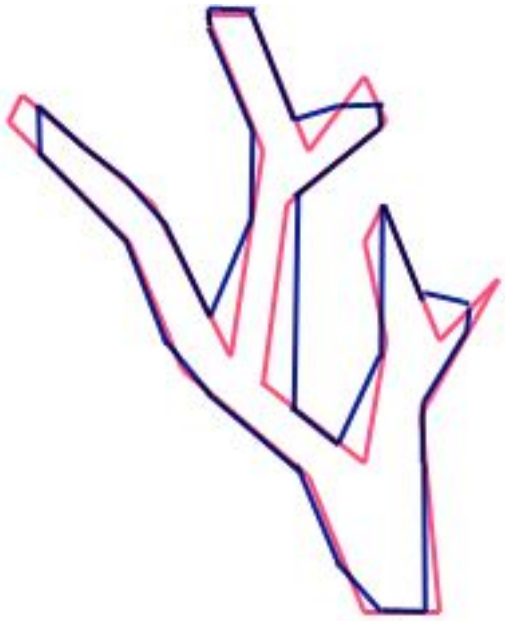
Sanal heykeltıraşık sistemlerinde obje dış çeper ile ifade edildikten sonra işleyici alet ile işlenerek modellenir ve daha sonra yeniden inşa edilirler. Şekil 5.3.2'deki nesne dexellerle ifade edildikten sonra herhangi bir işleme veya modellemeye uğramadan yeniden inşa edilirse Şekil 5.3.3'deki gibi hatalı bir yapıya dönüştürülecektir.



Şekil 5.3.3: Dexeller ile İfade Edilen Objenin Yeniden İnşası

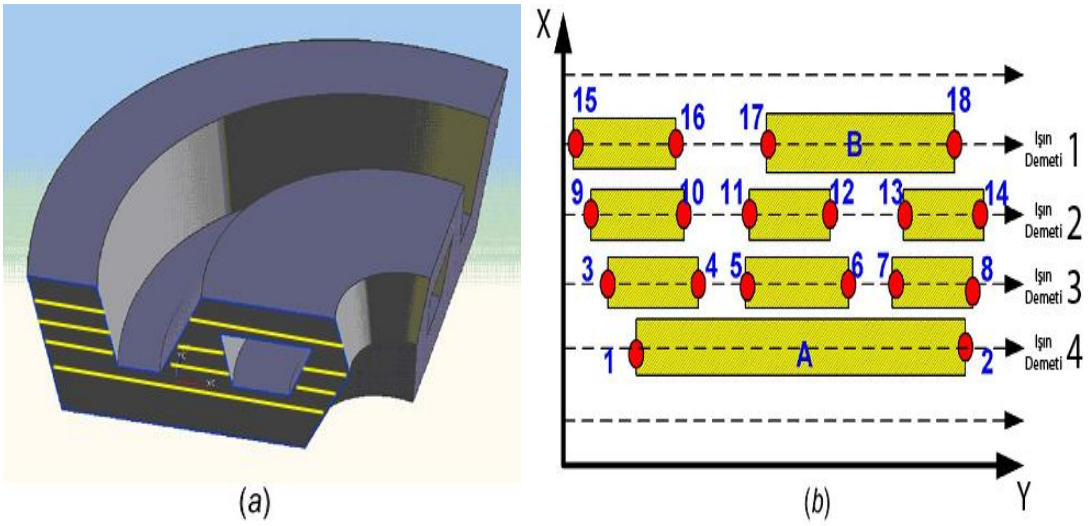
Şekil 5.3.3'de tam olarak farkedilemeyen hata, Şekil 5.3.4'te çok daha net bir şekilde görülebilir. Mavi renkteki model yeniden inşa edildikten sonra elde edilmişken, penbe renkli model, ilk baştaki orijinal modeldir. Dikkat edilirse, özellikle modelin keskin hatlara sahip olduğu kısımlarda dexeller daha kötü sonuç vermiştir.

Son kullanıcılara gerçeğe daha yakın sonuçlar vermeyi hedefleyen sanal heykeltıraşılık sistemleri için bu tarz sorunlar kabul edilemezdir.



Şekil 5.3.4: Orijinal Model ile Yeniden İnşa Edilen Model Arasındaki Fark

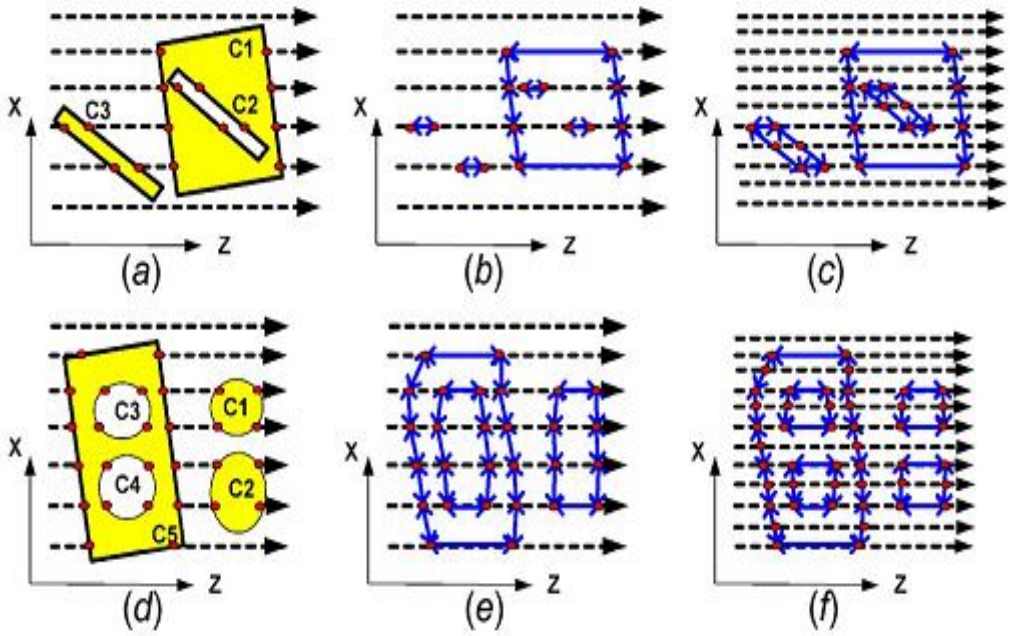
Yeniden inşa esnasında da bu tarz birçok problemle karşılaşılmaktadır. Karmaşık bir şekil üzerinde bir inceleme yapılacak olursa anomalilerin algoritmalar üzerindeki etkileri daha anlaşılır bir şekilde ortaya konacaktır.



Şekil 5.3.5: Yeniden İnşa Anomalisi

Şekil 5.3.5(a)'daki karmaşık nesne modeli üzerine 4 adet ışın demeti gönderilsin ve ışın-düzlem testi ile dexeller elde edilmiş olsun. Tez çalışması dış çeperlerden ziyade, yeniden inşa üzerine yoğunlaştığından X ve Z koordinatları yerine X-Y düzlemi üzerindeki kesitler incelenecektir. Şekil 5.3.5(b)'ye bakıldığında Y düzlemi boyunca model üzerinden X ekseninde alınmış kesitler görülmektedir. Yeniden inşa aşamasındaki problem Işn demetleri ile kesişim sonucu elde edilen kesitlerin X ekseni doğrultusunda birbirileriyle ne şekilde örülerek birleşecekleri konusudur.

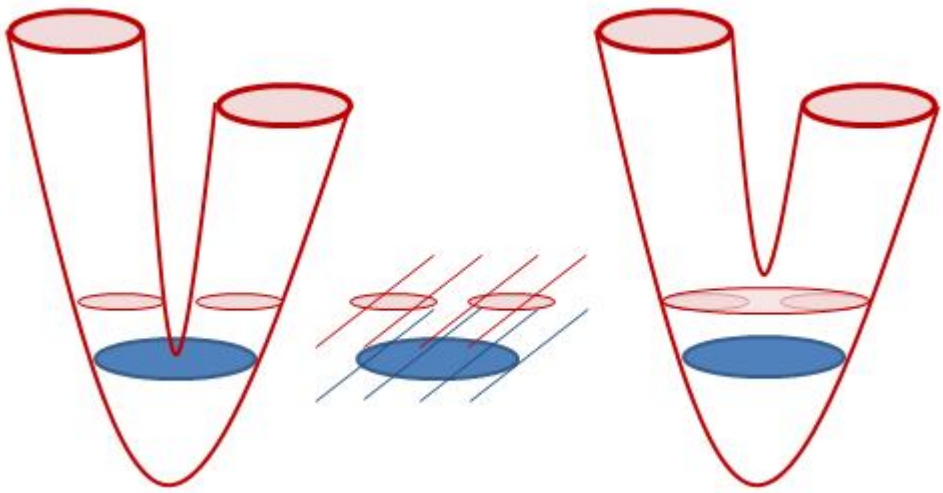
Örneğin Şekil 5.3.5 (b)'de en önemli sorun, 3. Işın demeti üzerindeki 4 ve 5. Noktalar ile 6 ve 7. noktaların hemen bir altta 4. Işın Demetindeki hangi nokta ile birleşmeleri gerektiği sorunudur. Bu tarz problemleri aşmak için yeniden inşa algoritmalarına, anomalileri aşacak birtakım kıstaslar getirilmelidir. Aksi halde modelin işlenmesi sırasında modelin geometrik yapısı üzerine uygulanan bütün güncellemeler yetersiz bir şekilde son kullanıcıya aktarılacaktır.



Şekil 5.3.6: Anomali Örnekleri

Şekil 5.3.6'ya bakıldığında (a) ve (d)'de orijinal şekilleri, (b) ve (e)'de hatalı yeniden inşa metotlar, (c) ve (f)'de ise olması gereken çözüm görülmektedir. Önerilen metotta bu eşleştirmeler normal ve normale dik vektörlerin X ekseni boyunca birbirlerine olan konumlarına ve yönlerine göre yapılmaktadır.

Normal vektörlerinin konumu bir alttaki kesitte hangi nokta ile eşleşileceği hakkında önemli ipuçları vermektedir. Bu durum yalnızca bir alt seviyedeki çeper üzerinde bulunan noktalar için değil, aynı zamanda bir üst seviyedeki noktalar içinde geçerlidir. Normal vektörlerinin, modelin inşasında ne kadar etkili olduğunu Şekil 5.3.7 ile daha iyi görebiliriz.



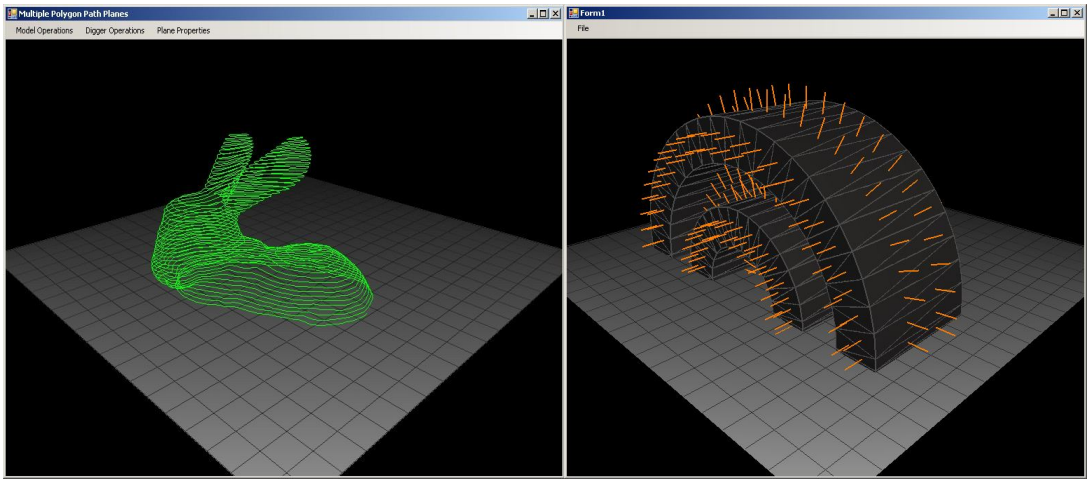
Şekil 5.3.7: Normal Vektörlerinin Önemi

Şekilde görülen sol taraftaki nesnenin orijinal durumu, normal vektörleri kullanılmadığında sağ tarafta görülen model haline dönüşmektedir. Bu da nesnenin yeniden inşasında büyük bir problem teşkil etmektedir. Orta kısımda temsil edilen gösterim, normal vektörlerinin ard arda sıralanmış dış çeperlerin birbirleri ile nasıl eşleşeceğine dair ipuçları verdiğini daha açıklayıcı şekilde yansıtmaktadır.

Modellenen nesneler yeniden inşa aşamasına geldiklerinde temsil edildikleri dış çeperlerin birbirleri ile nasıl birleşecekleri, tamamen kendilerini oluşturan noktalar üzerinde tanımlanan normal ve normale dik vektörlerin birbirlerine göre konumları doğrultusunda hesaplanmaktadır.

Anomaliyi oluşturan en önemli sorunun ise iki adet çember ile temsil edilmiş dış çeperin bir altta tek bir mavi daire ile gösterilmiş dış çeper ile birleşmesi gerektiği için kaynaklandığı görülmektedir. Sağ tarafta, algoritma bu durumu tespit edemediğinden üstteki iki çemberlik dış hattı, tek bir çember olarak ele almış ve alttaki tek çemberlik dış çeperle eşleştirmiştir. Normal vektörleri bu gibi durumlarda kıstas oluşturarak devreye girmektedir.

Sanal düzlemlerde ise yatay düzlemde sıklık problemi yaşanmamaktadır. Çünkü her bir sanal düzlem doğrudan nesne ile tümüyle bir kesişime tabi tutulduğundan bütün modellerde kusursuz sonuç vermektedir. Dikey doğrultuda sanal düzlemlerin sıklığı ise dış çeperlerin arasının doldurulmasındaki doğruluk payının artmasını sağlayacaktır (Şekil 5.3.8).



Şekil 5.3.8: Sanal Düzlemlerin Modeller ile Kesişimi

6. SONUÇ

Yapılan çalışma ile, sanal heykeltıraşlık sistemlerine alternatif bir yaklaşım içerisinde gerek dallanma(dallanma) gerekse de dış hat belirleme(contouring) problemlerinin etkin bir çözümü ile tam bir sistem önerilmiş ve gerçeklenmiştir.

OpenGL grafik kütüphanelerinin ve .Net platformu üzerindeki C# programlama dilinin zenginliklerinden yararlanılarak geliştirilen uygulama, çalışma ortamına aktarmış olduğu üç boyutlu modeller üzerine sanal düzlemleri kesiştirerek, matematiksel hesaplamalarla hızlı bir şekilde dış hat çeperlerini bulmakta ve dış çeperlerle tanımladığı objeler üzerinde parça koparma ve ekleme gibi sanal heykeltıraşlık işlemleri gerçekleştirerek, normal ve normale dik vektörlere bağlı olarak bir takım kıstaslar altında yeniden inşa etmektedir.

Elde edilen dış hatlar, yeni bir örgü algoritması kullanılarak yine bazı kıstaslar altında örülerek kapalı bir hacme ve katı görünüme geçirilmiştir. Dış çeperlerin üçgen ilkelleri ile örülmesi işlemi esnasında yapılan hesaplamalarda, dış çeper poligonlarını oluşturan her doğru parçasının başlangıç ve bitiş noktalarındaki normal ve normale dik olan vektörlerin birbirlerine göre konumları esas alınmış ve ilgili kıstaslar bu vektörlere göre tanımlanmıştır.

Normal ve normale dik vektörlerin birbirlerine göre konumları, dış çeperlerin hem doğru bir şekilde eşleşmelerinin yapılmasını, hem de hesaplamaların daha hızlı olmasını sağlamıştır. İşleme sürecine geçmeden, poligonu oluşturan doğru parçalarının başlangıç ve bitiş noktalarının koordinatlarının tutulması ve görüntü listelerinin kullanımı hızlanmayı etkileyen ve sağlayan diğer önemli faktörler olarak göze çarpmaktadır.

Dış hatlar arası boşluğun örülmesinde en sık görülen sorun, çeperler arası örgüleme işlemleri esnasında yanlış noktalar arasında eşleştirme yapılarak hatalı modeller ortaya çıkarmaktır. Bu tez çalışmasında bu tür hatalardan arınmak için normaller ve normallere dik olan vektörler kullanılmış ve doğru eşleştirme yapılması amaçlanmıştır.

Karmaşıklık, sistem kaynaklarının tüketimi ve elde edilen çeperlerin yumuşak hatlara sahip olmaması gibi problemler hem dış hat algoritmalarında hem de yeniden inşa algoritmalarında görülen başlıca sorunlardır. Düzlem kullanılarak algoritmanın karmaşıklığı giderilmiş ve sistem kaynaklarının kullanımı da aynı derece de düşürülmüştür. Düzlemlerin sık aralıklarla gönderilmesinin yanında, elde edilen dış hatlara birtakım eğri(spline) algoritmaları uygulanarak oluşan yapının daha keskin bir görüntüye sahip olması sağlanabilir.

Kolay ve dinamik veri yapısının esnekleştirilmesi, algoritma içinde saklanan modelin geometrik yapısına ait bilgilerin sanal heykeltıraşlığın diğer aşamalarında kullanılabilecek şekilde işlenmesi ve algoritmanın bu aşamalar için optimize edilmesi, önerilen metot için yapılması düşünülen çalışmalardır.

KAYNAKLAR

[1] **Zhu, W. and Lee, Y.**, 2004, "Dexel-based force–torque rendering and volume updating for 5-DOF haptic productprototyping and virtual sculpting, *Computers in Industry"*, **55**, *125–145*.

[2] **Mark, W., Randolph, S., Finch, M., Van Verth, J., Taylor RM**, 1996, "Adding force feedback to graphics systems; issues and solutions", *Computers Graphics (Proc. SIGGRAPH '96)*, 447-452

[3] **Zhu, W., Lee, Y.,** 2005, "A Visibility Sphere Marching Algorithm of Constructing Polyhedral models for haptic sculpting and product prototyping", *Robotics and Computer-Integrated Manufacturing*, **21(1)**, 19-36.

[4] **Raffin, R., Gesquière, G., Remy, E., Thon, S.**, 2004, "VirSculpt: A virtual sculpting environment", *International Conference Graphicon.*

[3] **Leu, M.C. and Peng, X.,** 2003, "Virtual Sculpting with Haptic Interface", NSF design, service and manufacturing grantees and research conference.

[4] **Dachille, F., Qui, H. and Kaufman, A.**, 2001, "A Novel Haptics-Based interface and sculpting system for physics-basedgeometric design", *Computer-Aided Design*, **33**, 403-420.

[5] **Leu, M.C., Zhang, W.,** 2008, "Virtual Sculpting with Surface Smoothing Based on Level Set Method", *CIRP Ann.*, **57(1)**, 167-170.

[6] **Galyean, T.A., Hughes, J.F.,** 1991, "Sculpting: An interactive volumetric modeling technique", *Computer Graphics (Proc. SIGGRAPH '91)*, **25(4)**, 267–274.

[7] **McDonnell, K.T., Qin, H., Wlodarczyk, R.A.,** 2001, "Virtual Clay: A real time sculpting system with haptic toolkits", ACM Symposium on Interactive 3D Graphics *(Procceedings of the 2001)*, 179-190.

[8] **Mizuno, S., Okada, M., Toriwaki, J.,** 1998, "Virtual sculpting and virtual woodcut printing", *The Visual Computer*, **14(2)**, 39-51.

[9] **Galyean, T.A., Hughes, J.F.,** 1991, "Sculpting: An interactive volumetric modeling technique", *Computer Graphics (Proc. SIGGRAPH '91)*, **25(4)**, 267–274.

[10] **Wang, W.S., Kaufman, A.E.,** 1995, "Volume Sculpting", *ACM Symposium on Interactive 3D Graphics*, 151-156.

[11] **Ferley, E., Cani, M.P., Gascuel, J.D.,** 1999, "Practical Volumetric Sculpting", *The Visual Computer*, **16(8)**, 469-480.

[12] **J. H. Ryu, H. S. Kim, K. H. Lee,** 2003, "Contour-based algorithms for generating 3D CAD models from medical images", *Int J Adv Manuf Technol*, 112–119.

[13] **Kobbelt, L. P., Botsch, M., Schwanecke, U., Seidel, H.-P.,** 2001, **"**Feature Sensitive Surface Extraction from Volume Data", *Computer Graphics (SIGGRAPH 01)*, 57-66.

[14] **Ayasse, J., Müller, H.,** 2001, "Interactive manipulation of voxel volumes with free-formed voxel tools", *VMV 2001*, 21-23.

[15] **Boonma, A.,** 2006, “Haptic-Based Sharp Edge Retaining and Gap Bridging Algorithms for Computer Aided Design (CAD) and Reverse Engineering (RE)”.

[16] **Attene M., Falcidieno B, Rossignac J and Spagnuolo M. Edge-Sharpener**, 2003, “Recovering sharp features in triangulations of non-adaptively re-meshed surfaces”, *Proceedings of 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 62-69.

[17] **Chen C-Y. Cheng K-Y., Liao H.Y.M.,** 2005, “A Sharpness Dependent Approach to 3D Polygon Mesh Hole Filling”, *Proceedings of EuroGraphics 2005, Short Presentations*, 13-16.

[18] **Zhu, W.; Lee, Y.-S.,** 2005, “A marching algorithm of constructing polyhedral models from Dexel models for haptic virtual sculpting, *Robotics and Computer-Integrated Manufacturing*, **21(1),** 19-36**.**

[19] **Grossman, T., Balakrishnan, R., Singh, K**., 2003, "An interface for creating and manipulating curves using a high degree-of-freedom curve input device", *Proceedings of the conference on Human factors in computing systems*, Ft. Lauderdale, Florida, USA, 185-192.

[20] **Bitter, E., Nocent, O., Heff, A.,** 2004, “Spline and Ideal: From Real to Virtual Sculptures, and Back”, 001-007.

[21] **Ren, Y., Zhu, W., Lee, Y.,** 2008, “Feature Conservation and Conversion of Tri-dexel Volumetric Models to Polyhedral Surface Models for Product Prototyping”, *Computer-Aided Design and Applications*, 932-941.

[22] **Yau, H-T., Kuo, C-C., Yeh C-H.,** 2003, Extensino of surface reconstruction algorithm to the global stitching and repairing of STL models**,** *Computer-Aided Design***, 35(5)**, 477-486.